AFWAL-TR-82-3087
Volume II

AUTOMATED DESIGN OF DAMAGE RESISTANT STRUCTURES

Volume II - Program User's Manual

R. F. Taylor

University of Dayton Research Institute
Dayton, Ohio 45469

October 1982

Final Report for Period 1 June 1979 - 30 April 1982

FLIGHT DYNAMICS LABORATORY
AIR FORCE WRIGHT AERONAUTICAL LABORATORIES
AIR FORCE SYSTEMS COMMAND
WRIGHT-PATTERSON AIR FORCE BASE, OHIO 45433

NOTICE

When Government drawings, specifications, or other data are used for any purpose other than in connection with a definitely related Government procurement operation, the United States Government thereby incurs no responsibility nor any obligation whatsoever; and the fact that the government may have formulated, furnished, or in any way supplied the said drawings, specifications, or other data, is not to be regarded by implication or otherwise as in any manner licensing the holder or any other person or corporation, or conveying any rights or permission to manufacture, use, or sell any patented invention that may in any way be related thereto.

This technical report has been reviewed and is approved for publication.

VIPPERLA B. VENKAYYA
Project Engineer
Design & Analysis Methods Group

FREDERICK A. PICCHIONI, Lt Col, USAF
Chf, Analysis & Optimization Branch

FOR THE COMMANDER:

RALPH L. KUSTER, JR., Col, USAF
Chief, Structures & Dynamics Div.

"If your address has changed, if you wish to be removed from our mailing list, or if the addressee is no longer employed by your organization please notify AFWAL/FIBRA, W-P AFB, OH 45433 to help us maintain a current mailing list."

Copies of this report should not be returned unless return is required by security considerations, contractual obligations, or notice on a specific document.

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER AFWAL-TR-82-3087, Volume II	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) AUTOMATED DESIGN OF DAMAGE RESISTANT STRUCTURES Volume II - Program User's Manual		5. TYPE OF REPORT & PERIOD COVERED Final Report Covering Period From 6/1/79 - 4/30/82 6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) R. F. Taylor		8. CONTRACT OR GRANT NUMBER(s) F33615-79-C-3209
9. PERFORMING ORGANIZATION NAME AND ADDRESS University of Dayton Research Institute 300 College Park Avenue Dayton, Ohio 45469		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS 24010233
11. CONTROLLING OFFICE NAME AND ADDRESS Flight Dynamics Laboratory (AFWAL/FIBRA) Air Force Wright Aeronautical Laboratories(AFSC) Wright-Patterson Air Force Base, Ohio 45433		12. REPORT DATE October 1982 13. NUMBER OF PAGES 193
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)		15. SECURITY CLASS. (of this report) Unclassified 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

This report consists of two volumes. Volume I is titled, "Theory and Application."

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

structural analysis	stress analysis
vulnerability analysis	deflection analysis
structural optimization	finite element analysis
damage	automated design

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

This report documents an effort to develop an optimality criterion approach to the design of damage tolerant structures subject to stress, deflection, and frequency requirements. Damage conditions are treated in an integral manner in the resizing algorithm. An iterative reanalysis procedure is used to improve the efficiency of the static analyses that are needed as the optimization proceeds. In this volume of the report,

(Block 20 Continued)

input instructions to the computer code ADDRESS (Automated Design of Damage Resistant Structures) are detailed. Further information is included which shows the required control card sequence, the program structures, and subroutine descriptions. The program is operational on the CDC CYBER system at Wright-Patterson Air Force Base.

FOREWORD

This final report documents work performed by the Aerospace Mechanics Division of the University of Dayton Research Institute (UDRI) for the Structures and Dynamics Division of the Flight Dynamics Laboratory, Air Force Wright Aeronautical Laboratories (AFWAL), Wright-Patterson Air Force Base, Ohio. The work was performed under contract F33615-79-C-3209. Dr. V. B. Venkayya was the AFWAL Project Engineer.

This report consists of two volumes. Volume I, entitled, "Theory and Application," describes the theory behind the design optimization method and gives design results. In Volume II, Program User's Manual," detailed instruction are given for use of the ADDRESS (Automated Design of Damage Resistant Structures) computer code on the Wright-Patterson Air Force Base CDC computing system. The report covers work conducted between 1 June 1979 and 30 April 1982.

Dr. Ronald F. Taylor was the Principal Investigator and Dr. Fred K. Bogner was the Project Manager. The author acknowledges the important contributions of Dr. Bogner and Dr. Robert A. Brockman during the formative stages of the program. Acknowledgement is also made of the programming assistance of Mr. Keith Miller and Mr. Jerry Jensen.

TABLE OF CONTENTS

SECTION		PAGE
1	INTRODUCTION	1
	1.1 OVERVIEW	1
	1.2 ORGANIZATION OF THE SUPPLEMENTAL DATA	1
2	INPUT INSTRUCTIONS FOR THE ADDRESS PROGRAM	3
3	PROGRAM OUTPUT	23
4	CONTROL CARDS AND PROGRAM UPDATES	27
5	PROGRAM ORGANIZATION	30

APPENDICES

A	VARIABLE NAME / SUBROUTINE NAME CROSS-REFERENCE TABLE	39
B	COMMON BLOCK NAME / SUBROUTINE NAME CROSS-REFERENCE TABLE	65
C	SUBROUTINE DESCRIPTIONS	70
D	UPDATES FOR OUT OF CORE SOLUTIONS	171
E	TRUSS PROGRAM LISTING	177

REFERENCES 193

LIST OF ILLUSTRATIONS

FIGURE		PAGE
1	ADDRESS Program Routines: MIAN, DØPRØB, and INITIAL Call Sequence.	33
2	Input Routines: INITP Call Sequence.	34
3	Analysis Routines: CURRENT Call Sequence.	35
4	Displacement Resizing Routines: DMØDE Call Sequence.	36
5	Print Routines: PRINT Call Sequence.	37
6	Analysis Routine Segments E, F, G, and H.	38

LIST OF TABLES

TABLE		PAGE
1	Control Card Example without Segmentation	27
2	Control Card Example with Segmentation	28

SECTION I
INTRODUCTION

In this report user's information for the ADDRESS program (Automated Design of Damage Resistant Structures) is presented. The first volume of this report discusses the theory behind the ADDRESS program and presents representative applications. In this introductory section an overview of the user's instruction manual is given. Also included is a discussion of the supplemental data found in the appendices.

1.1 OVERVIEW

Section 2 of this report details the data input requirements. Particular attention should be paid to the user notes at the end of that section.

Program output is discussed in Section 3. This includes printed output as well as informaton to local files for further processing. Formats for these files are specified in the code.

Control cards and program updates are given in Section 4. Several distinct versions of ADDRESS can be easily obtained to handle vibration problems and models which use beam finite elements. Segmentation directives are also given which help to reduce the core requirements for analysis problems.

The organization of the subroutines is detailed in Section 5. This information is helpful in adding new features to the program.

1.2 ORGANIZATION OF THE SUPPLEMENTAL DATA

In the appendices other useful programing information are presented. Appendices A and B were generated using the UDRI cross-reference program XREF[2]. The first two pages of Appendix A contain a brief explanation of the use of XREF.

The variable name - subroutine name cross-reference table of Appendix A gives an alphabetical listing of all the program variables. After the colon following the variable name is a listing

of all the subroutines where this variable is used. An asterisk before the subroutine name indicates that the variable is defined or redefined in that routine. Appendix B is a list of the common blocks and the routines in which they appear.

The subroutine descriptions of Appendix C provide the following information:

- subroutine name (the descriptions are in alphabetical order by name);
- a short statement of the purpose of the routine;
- the call sequence with the arguments as they appear in the subroutine;
- a description of the arguments;
- a list of the subroutines which call the given subroutine;
- a list of the externals, including any intrinsic functions;
- the files used by this routine;
- special notes or user information.

The updates in Appendix D show the changes that are needed to put the mass and stiffness matrices out of core. The PUTSK, PRNTSK, and SK function routines are listed with extensive user comments.

Appendix E is a listing of the TRUSS program discussed in Reference 1 which compares various iterative reanalysis techniques for the calculation of stresses and deflections in a damaged truss structure. The program is written for the CDC Fortran V compiler.

SECTION 2

INPUT INSTRUCTIONS FOR THE ADDRESS PROGRAM

In this section the data requirements of the ADDRESS program are discussed. The input consists of the following blocks of data:

A. General Input

B. Coordinate Data

C. General Element Data

D. Materials Data

E. Connectivity Data

F. Fiber Orientation Data (optional)

G. Boundary Conditions

H. Loads Data

I. Displacement Data (optional)

J. Damage Data (optional)

K. Lumped Mass Data (optional)

The following pages describe each of these data blocks in detail. Each data block consists of one or more card sets, and each card set consists of one or more cards. The notes inform the user as to the number of cards in each set as well as explanatory information.

A. GENERAL INPUT-READ IN SUBROUTINE INITIAL (CARD SET 1) AND INGNRL (CARD SETS 2-5)

CARD SET (FORMAT)	VARIABLE NAME	DESCRIPTION	NOTES
1 (1I5)	NSTR	Number of problems to be solved	-
2 (8A10)	TITLE	Any alphanumeric description of the problem	-
3 (16I5)	MM	=2 for a two-dimensional problem =3 for a three-dimensional problem	-
	LMTDSP	=0 if there is no displacement constraint =1 if displacement constraint is the same for all nodes =2 if displacement constraint is not the same for all nodes	-
	LMTSTR	=0 if no stress constraints exist =1 if stress constraints exist	-
	NDMGCAS	Number of damage cases	-
	NLMPMSS	Number of lumped masses	-
	MAXDCCL	Number of cycles of iteration using the recursion relation based on displacement gradients	-
	MAXECCL	Number of cycles of iteration using the recursion relation based on energy gradients	-
	IAREAS	=0 if the initial design variables of the elements are set to 1.0 inches =1 if you wish to input the initial design variables for each element	-
	INDMIN	=0 if the minimum allowable size is the same for all elements =1 if the minimum sizes of the elements are to be input directly	-

CARD SET (FORMAT)	VARIABLE NAME	DESCRIPTION	NOTES
	KANLYZE	=0 if you wish to use the program for structural optimization =1 if you wish to use the program for structural analysis only (no resizing)	-
	MAXSZE	=0 if no maximum size is to be specified for the elements =1 if the maximum allowable sizes of the elements are to be input directly	-
	LAYERD	=0 if problem contians no layered composite elements =1 if problem contains layered composite elements	-
	INDANG	=0 if no composite elements or 0° fibers are defined per elements with respect to the global coordinate system =1 if the 0° fibers are defined per element with respect to the local coordinate system =2 if 0° fiber orientation is the same for all composite elements with respect to the global coordinate system	-
	MNLAYR	=0 if same for composite elements =1 if the minimum proportions of 0°, 90° ±45° layers will be input for each member	-
	IPP	=0 if no postprocessor file present =1 if postprocessor file on DPOST created	-
4 (8F10.3)	AEMNMM	Minimum allowable design value	-
	DINCR	A parameter to determine the active set of displacement constraints	-
	THKLAM	Minimum composite layer thickness	-
	SPRDF	Shear panel reduction factor	

CARD SET (FORMAT)	VARIABLE NAME	DESCRIPTION	NOTES
5 (2I5, F10.0)	MODES	Number of eigenvalues to compute	-
	NOI	Maximum number of iterations	-
	TOLVEC	Eigenvector tolerance	-

B. COORDINATE DATA - READ IN SUBROUTINE INXYZ

CARD SET (FORMAT)	VARIABLE NAME	DESCRIPTION	NOTES
6 (15X,2I5)	NODES	Total number of nodes in the finite element model. Limit is 90.	-
	MGEN	Mesh generator flag =0 if you wish to read in the coordinates =1 if the coordinates are to be read or generated in the user subroutine MESG	-
7 (I5,A1,I4 3E10.0)	N	Node Number	(1)
	ISYS	Reference coordinate system =_: Cartesian system x,y,z, =A: Cylindrical R,θ,Z =B: Spherical R,θ,φ =C: (user defined) =D: (user defined) =E: (user defined)	-
	NINCR	Increment for node point generation	(2)
	X1	X-node coordinate	-
	X2	Y-node coordinate	-
	X3	Z-node coordinate	(3)

C. GENERAL ELEMENT DATA - READ IN SUBROUTINE ELEMIN

CARD SET (FORMAT)	VARIABLE NAME	DESCRIPTION	NOTES
8 (4I5)	ITYPE	Always=3 (fixed variable)	(4)
	NMT	Total number of material property sets. Limit is 20.	-
	NELEM	Total number of elements	-
	NCØMP	Number of composite material property sets. Must always be less than or equal to NMT	-

D. MATERIALS DATA - READ IN SUBROUTINE INPO3

CARD SET (FORMAT)	VARIABLE NAME	DESCRIPTION	NOTES
9 (5E10.0)	YØUNGM(I)	If material is isotropic (same elasticity in all directions) then it is the Young's modulus for the I-th material (psi) If material is composite then it is the elastic modulus in the 0° fiber direction for the I-th material	(5)
	PØISØN(I)	If the material is isotropic then it is Poisson's ratio for the I-th material If the material is composite then it is Poisson's ratio for the transverse strain due to stress along the 0° fiber direction for the I-th composite material	-
	RHØ1(I)	Density of the I-th material	-
	ELASM(I)	If the material isotropic then =0 If material is composite then it is the elastic modulus transverse to the 0° fiber direction of the I-th composite material (psi)	-
	SHEARM(I)	If material is isotropic then =0 If material is composite then it is the shear modulus for the I-th composite material	-
10 (5E10.0)	ALSTRS(1)	Tension stress allowable in psi in the 0° fiber direction	(5)
	ALSTRS(2)	Compression stress allowable in psi in the 0° fiber direction	
	ALSTRS(3)	Tension stress allowable in psi transverse to the 0° fiber direction	

CARD SET (FORMAT)	VARIABLE NAME	DESCRIPTION	NOTES
	ALSTRS(4)	Compression stress allowable in psi transverse to the 0° fiber direction	-
	ALSTRS(5)	Shear stress allowable in psi transverse to the 0° fiber direction	-

If LAYERED (see Card Set 2) = 0, go to Card Set 18

If INDANG (see Card Set 2) = 1, go to Card Set 13

If INDANG (see Card Set 2) = 2, go to Card Set 14

If ITYPE ≠ 1, omit Card Sets 11A and 11B.

E. CONNECTIVITY DATA - READ IN SUBROUTINE INCØNN

CARD SET (FORMAT)	VARIABLE NAME	DESCRIPTION	NOTES
11 (9I5, 3E10.0)	IELNO	Element number	(6)
	ITYPE	Element type =1 for beam element =2 for axial bar =3 for triangular membrane =4 for quadrilateral membrane =5 for shear panel	-
	IPR	Material property set for this element	(7)
	LAMFØ	Fiber orientation parameter =0 for isotropic element =1 for fiber orientations 0°, 90°, ±45° in the proportions .25, .25, .50 respectively. =2 for fiber orientations 0°, 90° in the proportions .50, .50 respectively. =3 for fiber orientations ±45° in the proportion 1.00 =4 for fiber orientations 0°, ±45° in the proportions 1/3, 2/3 respectively. =5 for fiber orientations 90°, ±45° in the proportions 1/3, 2/3 respectively.	-
	KGEN	Node increment for element generation	(8)
	NØD(1) NØD(2) NØD(3) NØD(4)	Local node number 1 Local node number 2 Local node number 3 Local node number 4	(9)
	THICK	Member size	(10)
	THICKMN	Minimum element thickness	(10)
	THICKMX	Maximum element thickness	(10)

CARD SET (FORMAT)	VARIABLE NAME	DESCRIPTION	NOTES
11A (5X, 7E10.0)	YI1	Moment of inertia about the y-axis at node 1	-
	YI2	Moment of inertia about the y-axis at node 2	-
	ZI1	Moment of inertia about the z-axis at node 1	-
	ZI2	Moment of inertia about the z-axis at node 2	-
	XJ1	Polar moment of inertia at node 1	-
	XJ2	Polar moment of inertia at node 2	-
	YK1	y-component of the mass moment of inertia at node 1	-
11B (5X, 7E10.0)	YK2	y-component of the mass moment of inertia at node 2	-
	ZK1	z-component of the mass moment of inertia at node 1	-
	ZK2	z-component of the mass moment of inertia at node 2	-
	YM ZM	Mass centroid offsets	
	YG ZG	Geometric centroid offsets	

F. FIBER ORIENTATION DATA - READ IN SUBROUTINE INLAYR

CARD SET (FORMAT)	VARIABLE NAME	DESCRIPTION	NOTES
12 (3F10.3)	XANG(I)	The angle in degrees that the 0° fibers of the I-th element makes with the global x-axis	(11)
	YANG(I)	The angle in degrees that the 0° fibers of the I-th elelment makes with the global y-axis	-
	ZANG(I)	The angle in degrees that the 0° fibers of the I-th element makes with the global z-axis	-

GO TO CARD SET 16

CARD SET (FORMAT)	VARIABLE NAME	DESCRIPTION	NOTES
13 (6F10.3)	XANG(I)	The angle in degrees that the 0° fibers of the I-th element makes with the local element coordinate system	(12)

GO TO CARD SET 16

CARD SET (FORMAT)	VARIABLE NAME	DESCRIPTION	NOTES
14 (3F10.3)	XA	The angle the 0° fibers make with the global x-axis	(13)
	YA	The angle the 0° fibers make with the global y-axis	-
	ZA	The angle the 0° fibers make with the global z-axis	-

IF IAREAS = 0 SKIP CARD SETS 15 and 16

CARD SET (FORMAT)	VARIABLE NAME	DESCRIPTION	NOTES
15 (3F10.3)	AEX(I)	Proportion of fibers in the 0° direction for the I-th element	-
16 (3F10.3)	AEY(I)	Proportion of fibers in the 90° direction for the I-th element	-

IF MNLAYR = 0 SKIP CARD SET 17

17 (3F10.3)	AEXMIN(I)	Minimum proportion of 0° layers for the I-th element	-
	AEYMIN(I)	Minimum proportion of 90° layers for the I-th element	-
	AEXYMIN(I)	Minimum proportion of ±45° layers for the I-th element	-

G. BOUNDARY CONDITIONS - READ IN SUBROUTINE GETBC

CARD SET (FORMAT)	VARIABLE NAME	DESCRIPTION	NOTES
18 (2I5)	NBC1	Number of type 1 constraints	(14)
	NBC2	Number of type 2 constraints	(15)

IF NBC1 = 0 GO TO CARD SET 21

CARD SET (FORMAT)	VARIABLE NAME	DESCRIPTION	NOTES
19 (3I5)	N1	Beginning node number to be constrained	(16)
	N2	Ending node number to be constrained	
	INCR	Node number increment	
20 (3I5)	JD(I)	Nodal components constrained (type 1)	(16)-(17)

IF NBC2 = 0 SKIP CARD SETS 21 and 22

CARD SET (FORMAT)	VARIABLE NAME	DESCRIPTION	NOTES
21 (3I5)	JD(I)	Nodal components constrained (type 2)	(17)-(18)
22 (10I5)	ND(I)	Nodes constrained	(18)-(19)

H. LOADS DATA - READ IN SUBROUTINE INLOADS

CARD SET (FORMAT)	VARIABLE NAME	DESCRIPTION	NOTES
23 (16I5)	LOADS	Number of load conditions	-
	NJLOADS	Number of load components in the Ith loading condition	(20)
24 (3(F10.0, 2I5))	TEMP(I)	Magnitude of the Ith load	(21)
	IM(I)	Direction of the load =1 if in x direction =2 if in y direction =3 if in z direction	-
	JM(I)	Node number where load is applied	-

IF LMTDSP = 0 SKIP CARD SETS 25, 26, and 27

IF LMTDSP = 2 GOT TO CARD SET 26

I. DISPLACEMENT CONSTRAINT DATA - READ IN SUBROUTINE INDSPL

CARD SET (FORMAT)	VARIABLE NAME	DESCRIPTION	NOTES
25 (3F10.3)	DEFMAX(I)	Absolute value of the displacement constraint in the Ith direction for all nodes =1 for x direction =2 for y direction =3 for z direction	(22)

IF LMTDSP = 1 SKIP CARD SETS 26 and 27

CARD SET (FORMAT)	VARIABLE NAME	DESCRIPTION	NOTES
26 (I5)	KH	Number of displacement constraints	-
27 (3(F10.0, 2I5))	TEMP(I)	Magnitude of the displacement constraint	(23)
	IM(I)	Direction in which the constraint is applied =1 if in x direction =2 if in y direction =3 if in z direction	-
	JM(I)	Number of the node where constraint is applied	-

IF NDMGCAS = 0 SKIP CARD SETS 28, 29, and 30

J. DAMAGE DATA - READ IN SUBROUTINE INDMGE

CARD SET (FORMAT)	VARIABLE NAME	DESCRIPTION	NOTES
28 (2(F10.0, I5))	TOLDMGD	Damage displacement iteration tolerance	-
	MAXDDIT	Maximum number of damage displacement iteration	-
	TOLDMGF	Damage mode shape iteration tolerance	-
	MAXDFIT	Maximum number of damage mode shape iteration	-
29 (I5)	NDMG(J)	Number of damage elements in damage case J	(24)
30 (I5, 2F10.0)	IDNG(J,I)	Ith element in damage case J	-
	DKF	Stiff damage factor	-
	DMF (TR(J,I))	Mass damage factor in the Ith element and in the Ith damage case	-

IF NLMPMSS = 0 THEN STOP

K. LUMPED MASS DATA - READ IN SUBROUTINE INPMSS

CARD SET (FORMAT)	VARIABLE NAME	DESCRIPTION	NOTES
31 (I5, F10.0)	LMNØDE(I)	Node location of Ith lumped mass	(25)
	WLMPMSS(I)	Value of Ith lumped mass in lbf.	-

NOTES:

(1) Repeat Card Set 5 NØDES times, then add one blank card.

(2) A nonzero value of NINCR causes nodes to be equally spaced between the last and current nodes, with numbering increment NICR. As an example, the data

10		0.	0.	0.
20	2	10.	-10.	0.

is equivalent to

10	0.	0.	0.
12	2.	-2.	0.
14	4.	-4.	0.
16	6.	-6.	0.
18	8.	-8.	0.
20	10.	-10.	0.

(3) If MM = 2 (i.e., two-dimensional problem) then X3 is zero.

(4) ITYPE in ELEMIN is not the same as ITYPE in INPO3. In subroutine ELEMIN the following variable assignments are made:

NMAT = NMT
MEMBS = NELEM
ISØTRN = NMAT - NCØMP = Number of isotropic material property sets

(5) Repeat Card Sets 9 and 10 NMT times. The isotropic material properties are ordered before the composite properties.

(6) Repeat Card Set 11 NELEM times if KGEN = 0 (see Note 8 below). Add two blank cards at the end.

(7) This is determined by the ordering established in Card Sets 9 and 10.

(8) A nonzero value of KGEN on the second card of a pair causes elements between the last and current elements to be automatically generated. With the exception of node numbers, all elements generated are assigned the same data as the current element. Local node numbers for the previous element are incremented by KGEN to generate each succeeding element. More than one element must be generated to use this feature; node numbers for the current element need not be given. As an example, the data

85	5	1	0	0	49	50	52	51	.0190
88				2					

is equivalent to

85	5	1	0	0	49	50	52	51	.0190
86	5	1	0	0	51	52	54	53	.0190
87	5	1	0	0	53	54	56	55	.0190
88	5	1	0	0	55	56	58	57	.0190

(9) For beams, leave NØD(4) blank. Beam NØD(3) determines principal axes of bending.
For bars, leave NØD(3) and NØD(4) blank.
For triangular membrane, leave NØD(4) blank.
For each element, let NØD(1) be the lowest node number and NØD(2) the next lowest.
For quadrilaterals and shear panels, NØD(3) and NØD(4) are determined by continuing in the direction defined by NØD(1), NØD(2).

(10) For bars, THICK is the cross-sectional area. For all others, except beams, THICK is the element thickness.
For beam analysis, set THICK = 0; THICKMN = area at end A of beam; THICKMX = area at end B of beam.

(11) Include only if LAYERD = 1 and INDANG = 0 (see Card Set 3). Read continuously for I = 1 to I = NELEM. If an element is not composite, set XAND(I) = YANG(I) = ZANG(I) = 0.0 for that element.

(12) Include only if LAYERD = 1 and INDANG = 1 (see Card Set 3). Read continuously for I = 1 to I = NELEM. If an element is not composite set XANG(I) = 0.0 for that element.

(13) Include only if LAYERD = 1 and INDANG = 2 (see Card Set 3).

(14) A type 1 constraint is used to fix selected degrees of freedom over a specified range of nodes with specified increment in node numbers.

(15) A type 2 constraint is used to fix selected degrees of freedom over a list of nodes.

(16) Type 1 constraint data to be entered NBC1 times. Omit if NBC1 = 0.

(17) Values of 1, 2, and 3 can be used. To fix u at a node (x component of deflection) include a 1 value. To fix v at a node (y component of deflection) include a 2 value. To fix w at a node (z component of deflection) include a 3 value. For example, make Card Set 13

1 2 3

to fix all degrees of freedom. To fix only u use

1 0 0

(18) Type 2 constraint data to be entered NBC2 times. Omit if NBC2 = 0.

(19) Each of the nodes ND(I) are constrained in the components specified by Card Set 13; some of the ND(I) may be zero.

(20) Repeat LOADS times.

(21) Repeat for I = 1 to NJLØADS(I) for a total of (NJLØADS(I)/3 + 1) cards.

(22) Repeat MM times (see Card Set 3).

(23) Repeat for I = 1 to KH for a total of (KH/3 + 1) cards.

(24) Repeat NDMGCAS times (see Card Set 3).

(25) Repeat NLMPMSS times.

SECTION 3
PROGRAM OUTPUT

Output of the ADDRESS program is very similar in nature to the OPSTAT[3] program. Formats have been changed to some extent and additional information on damage cases is printed. Output for a typical static optimization run is as follows:

1. Input Data Echo. Fifty lines of data are printed per page with header and trailer numbers to help check the column locations of the data. The title card information is printed on all pages of the output, together with a page number.

2. General Data. Card Set 3 information is presented with explanations of the numerical values.

3. Nodal Coordinates. Data is always printed in Cartesian coordinates even though another system may be used for input.

4. Materials Data. Properties from Card Set 9 and the allowables from Card Set 10 are printed for each material.

5. Element Connectivity. Element number (ELEM), type (ITYPE), material code (MATL), fiber orientation parameter (LAM), local nude number, and member sizes are printed, based on the Card Set 11 data.

6. Composite Element Data. If the model contains composite elements the fiber orientation data from Card Sets 12, 15, 16, and 17 are printed.

7. Boundary Conditions. A summary of the type 1 and type 2 boundary conditions (see Notes 14 and 15 of Part G of Section 2) is printed. For the current library of elements, components 4 through 10 will always be printed as zeroes.

8. Load Summary. For each load case the net loads (FX, FY, FZ) in the coordinate directions are printed, together with their respective moments (MX, MY, MZ).

9. Damage Data. Data from Card Sets 28, 29, and 30 are printed for each damage case.

10. Population Information.[3] Output from Subroutine PØP concerning the distribution of elements in the stiffness matrix. This information is generated before the stiffness matrix of the structure is assembled.

(a) Gross Population = total number of elements in the upper triangle of the matrix.

Apparent Population = actual number of elements considered as non-zero by a given solution scheme. Thus the apparent population represents the number of storage locations required for the stiffness matrix.

(b) Starting Row Numbers for each column - the number of the row where the first non-zero element occurs in each column.

(c) Numbers of Diagonal Elements in Single Array Stiffness Matrix. For each Column I the actual number of elements, ID(I), in the upper triangular matrix up to and including that column, i.e.,

$$ID(I) = \frac{I(I+3)}{2} - \sum_{j=1}^{I} b_j$$

where b_j is the row number given for Column I in (b). Thus for the last column, ILAST,

ID(ILAST) = Apparent Population

11. Relative Design Data. After the static analysis is performed, the relative sizes of the elements are printed. The maximum relative size is 1.0 and all other sizes are given relative to this largest size. Next, a summary of the time for the analysis is printed. This is broken down into time for the matrix assembly, decomposition, and forward and backward substitution.

12. <u>Damage Reanalysis Error Summary</u>. If damage cases are included, then the analysis iteration error summary is printed for each case. The norm of the vector error is printed with the load case index in parentheses. Error information on the extrapolation process is also printed. Total time required to analyze each damage case is indicated. This should be less than the time for the analysis of Item 11 above for the reanalysis to be cost-effective.

13. <u>Initial Scale Factor (BASEA)</u>. This is defined in subroutine SCALE to be .01 times the square root of the quotient of STRMAX and ENGCAP. The variable STRMAX is the maximum over all load vectors of the inner product of the load and displacement vectors. In STIFFK, ENGCAP is defined as the sum over all elements of AE times ELENGTH where AE is the member size and ELENGTH is length (for bars) or area (for membranes and shear panels). The initial BASEA is used in PREPAR to scale the stress allowables. These scaled allowables, ALS, are used in STRCØN to compute the element stress ratios, ESRTIØ.

14. <u>Scaling Factors</u>. If a stress ratio for an element in some load/damage case combination exceeds 1.0, its maximum value is printed. The listing of corresponding element numbers that is printed beside these exceedance values tells the user which elements are overstressed for the design whose relative member sizes are scaled by the initial scaling factor of Item 13. The product of the initial scaling factor and the critical element scaling factors gives the new BASEA which is printed next with the header, "Scaling factor to satisfy stress constr."

15. <u>Summary of Current Cycle</u>. Current cycle number of this resizing is printed, together with a weights summary.

16. <u>Resizing Continues</u>. Items 11 through 15 are repeated until one of the following happens:

(a) All energy and displacement cycles have been completed as specified in Card Set 3;

(b) The weight goes up in an energy cycle;

(c) The weight doubles in a displacement cycle.

17. Final Design Stress Summary. Stresses, member sizes, element numbers, and ESRATIØ's are printed. The column of ESRATIØ's are the last set of ESRATIØ's computed (see Item 13). They are ordered so that the undamaged load cases come first, followed by the damage case information by load case. Example: 2 load cases, 2 damages,

	ELMT	ESRATIØ		
element no.	5	.91162E-01	LC1,	no damage
		.74404E-01	LC2,	no damage
		.12544E+00	LC1,	DC1
		.10595E+00	LC2,	DC1
		.91743E-01	LC1,	DC2
		.75008E-01	LC2,	DC2

LC = load case, DC = damage case.

18. Final Design Deflection Summary. Deflections are printed by node in the three coordinate directions. Applied loads are also printed in the same format as the ESRATIØ's above in Item 17.

Additional output is written to local files 8 and 99. The subroutines beginning with "PP" write information to one or both of these files for post-processing. See the subroutine descriptions in Appendix C for further information.

SECTION 4

CONTROL CARDS AND PROGRAM UPDATES

Table 1 lists the control cards needed to run the version of ADDRESS which is dimensioned for 453 elements (design variables) and 339 degrees of freedom. The central memory requirement of 255K octal can be reduced to 155K octal through the use of the control cards and segmentation shown in Table 2. When segmentation is used, the program capacity is doubled; however, the I/Ø requirement also doubles.

TABLE 1

CONTROL CARD EXAMPLE WITHOUT SEGMENTATION

```
UDNT,T3000,IO2000,CM255000.   D790486,TAYLOR,229-3018,93672
COMMENT.  INTERCOM BATCH JOB **** NO DECK********
ATTACH,LGO,DAMOPTBINARYJENSEN,MR=1.
ATTACH,TAPE5,NEWDATAFORMETALWING,CY=3,MR=1.
MAP(PART)
LIMIT,3100.
LGO(PL=10000)
REWIND,DEBUG.
COPY,DEBUG.
EXIT(S)
REWIND,DEBUG.
COPY,DEBUG.
```

TABLE 2

CONTROL CARD EXAMPLE WITH SEGMENTATION

```
UDST,T180,IO800,CM155000.   D790486,TAYLOR,229-3018,93672
COMMENT.   INTERCOM BATCH JOB **** NO DECK********
ATTACH,OLDPL,RFTDAMOPTPL.
ATTACH,C,RFTDAMOPTMODEZEROCHANGES.
COMMENT. THIS UPDATE CREATES THE MODES=0 DAMOPT
UPDATE,F,I=C,L=A1.
FTN,I=COMPILE,R=3.
COMMENT.    ATTACH,INPUT DATA FILE AS TAPE5
ATTACH,TAPE5,WINGDAMAGECASEECS,CY=2.
MAP,ON.
LDSET,PRESET=ZERO.
SEGLOAD.
LGO.
      TREE  DAMOPT-(DOPROB,INITIAL-ECHO)
      LEVEL
      TREE  AVERAGE
      TREE  SAVE
      TREE  EMODE
      TREE  GETBEST
      TREE  INITP-(POP,INPT-(TREE1,GETBC,INDMGE,INDSPL,INGNRL-PPTTL,
,INLAYR,INLOADS,INLPMSS,INXYZ-(CTYPE,MESHG,PPNODE)))
TREE1 TREE  ELEMIN-INPOS-INCONN-(NODCHCK,PPELEM)
      TREE  PRINT-(LAYPR-LAYCALC,PRNTDR-PPDISP,PRNTEL,PRNTMDS)
      TREE  DMODE-UNTFRC-UNITEG
      LEVEL
      TREE  CURRENT-(WEIGHTS,SIVIB2-(DECOUP,ERROR,ORTHOG,PREMULT,RANDOM)
,,SCALE-STRCON,DMGFREQ-(FRQITER-DMGORTH-PRELT),
,ANALYZ-(STIFFK-LLT-PRINTK,BOUND2))
CURRENT  INCLUDE BACKSUB,FORSUB
      LEVEL
      TREE  LMSIZE
      TREE  PPMODE-GENMSS
      TREE  RESTOR
      TREE  PPHDR
      TREE  REDUCE
      TREE  QDRLTL-(PLSTIF-CRAMER,CONDNS-CHANGE,SUM)
      LEVEL
      TREE  ELFORC
      TREE  DAMAGE-(DMGITER-DX,ERR,GETDK-ASEMDK)
      LEVEL
      TREE  TRNSFM
      TREE  COORD
      TREE  ELSTIC
      TREE  ELSTIF-LMPROD
      TREE  LMPMAS
      TREE  PLMASS
      TREE  PREPAR
      TREE  TRECON
      TREE  TRQDSTR-TRQDOUT-QLSTRS-STRESS
      COMMON
DMCLE  GLOBAL  DUMMY-SAVE
CURRENT  GLOBAL  GM-SAVE,SK-SAVE
      END DAMOPT
```

TABLE 2 (continued)

```
*ID DELETE1
*D RFTDAMOPT.52
      COMMON /GM/ GM(1)
*D RFTDAMOPT.236
      COMMON /GM/ GM(1)
*D RFTDAMOPT.2946
      COMMON /GM/ GM(1)
*D RFTDAMOPT.3707
      COMMON /GM/ GM(1)
*D RFTDAMOPT.555
      COMMON /EIGVEC/ U(350,6),V(350,6),W(1,1,1),XL(10,3),Y(1,1)
*D RFTDAMOPT.646
      COMMON /EIGVEC/ U(350,6),V(350,6),W(1,1,1),XL(10,3),Y(1,1)
*D RFTDAMOPT.718
      COMMON /EIGVEC/ U(350,6),V(350,6),W(1,1,1),XL(10,3),Y(1,1)
*D RFTDAMOPT.799
      COMMON /EIGVEC/ U(350,6),V(350,6),W(1,1,1),XL(10,3),Y(1,1)
*D RFTDAMOPT.1247
      COMMON /EIGVEC/ U(350,6),V(350,6),W(1,1,1),XL(10,3),Y(1,1)
*D RFTDAMOPT.1305
      COMMON /EIGVEC/ U(350,6),V(350,6),W(1,1,1),XL(10,3),Y(1,1)
*D RFTDAMOPT.1340
      COMMON /EIGVEC/ U(350,6),V(350,6),W(1,1,1),XL(10,3),Y(1,1)
*D RFTDAMOPT.1887
 5140 FORMAT(3F10.3)
*D RFTDAMOPT.2871
      COMMON /EIGVEC/ U(350,6),V(350,6),W(1,1,1),XL(10,3),Y(1,1)
*D RFTDAMOPT.3262
      COMMON /EIGVEC/ U(350,6),V(350,6),W(1,1,1),XL(10,3),Y(1,1)
*D RFTDAMOPT.3633
      COMMON /EIGVEC/ U(350,6),V(350,6),W(1,1,1),XL(10,3),Y(1,1)
```

SECTION 5

PROGRAM ORGANIZATION

The ADDRESS program contains 101 routines, counting the main program. Figures 1 through 6 show their interrelationship. In this section the overall organization of this calling sequence will be outlined. For details of the individual subroutines, see Appendix C.

The main program first calls INITIAL which sets certain variables and reads NSTR which is the number of problems to be solved in this run. The input data is echoed and mass storage is set up for the global mass and stiffness matrices. All of the analysis and optimization functions are controlled by DØPRØB, which is called within a loop whose index ranges from 1 to NSTR.

The DØPRØB routine shown in Figure 1 contains the program logic for performing all the aspects of the optimization. Its functions are as follows:

(1) Read the input data (call to INITP)

(2) Analyze the current design (call to CURRENT)

(3) Perform resizing for strength requirements (call to EMØDE)

(4) Perform resizing to meet displacement contraints (call to DMØDE)

(5) Perform bookkeeping functions such as getting the best design, averaging with a previous design, and saving the results (calls to GETBEST, AVERAGE, and SAVE)

(6) Print the results (call to PRINT)

During the input phase the routines of Figure 2 are called. In addition to input, a preliminary estimate of the storage space required for the mass and stiffness matrices is made by subroutine PØP. The input routines read the following types of data:

(1) General data such as analysis and optimization options to be performed (call to INGNRL);

(2) Coordinate data for the nodes in the finite element model (call to INXYZ);

(3) Element data including material properties, stress allowables, and connectivity (call to ELEMIN);

(4) Layer data for any composite elements (call to INLAYR);

(5) Boundary condition information (call to GETBC);

(6) Lumped mass data (call to INLPMSS);

(7) Applied load conditions (call to INLØADS);

(8) Information defining the damage conditions (call to INDMGE) and the displacement constraints (call to INDSPL).

All of the input data are also printed by these routines, and the model data is also put on the local file NFIL=99 for optional post-processing of the model.

Figure 3 shows the relationship of the various routines which perform the analysis of the current design. The functions performed by CURRENT are as follows:

(1) Set up the global mass and stiffness matrices and perform a static analysis (call to ANALYZ);

(2) Perform a vibration analysis (call to SIVIB2);

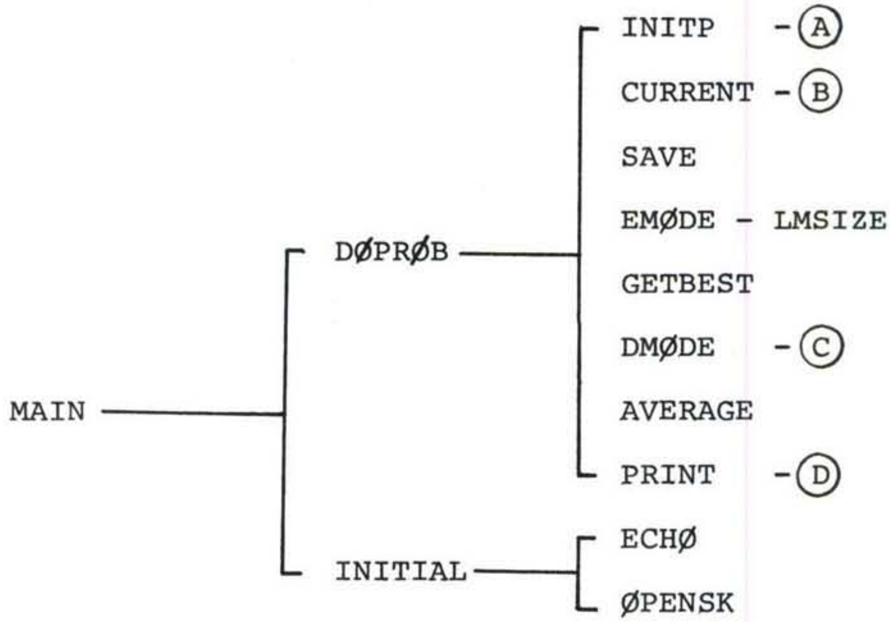
(3) For the damage conditions, compute the static (call to DAMAGE) and vibration (call to DMGFREQ) response by reanalysis methods;

(4) Scale the design to meet stress requirements (call to SCALE) and compute the weight (call to WEIGHT).

The routines shown in Figure 4 are called when resizing for displacement constraints takes place. Virtual displacements due to the dummy loads are determined by calls to REDUCE, FØRSUB, and BACKSUB. Calls to RESTØR and UNTFRC determine response to the

loads. Damage cases are handled by the call to DAMAGE. After resizing, LMSIZE is called to determine the thickness distributions for any composite elements that have been resized.

Figures 5 and 6 show the relationship of various utility routines. The print routines of Figure 5 involve calls to analysis routines to obtain results for the final design. The utility routines of Figure 6 set up the element mass and stiffnesses for both damaged and undamged elements.



Note: See Figures 2, 3, 4, and 5 for details of A, B, C, and D, respectively.

Figure 1. ADDRESS Program Routines: MAIN, DØPRØB, and INITIAL Call Sequence.

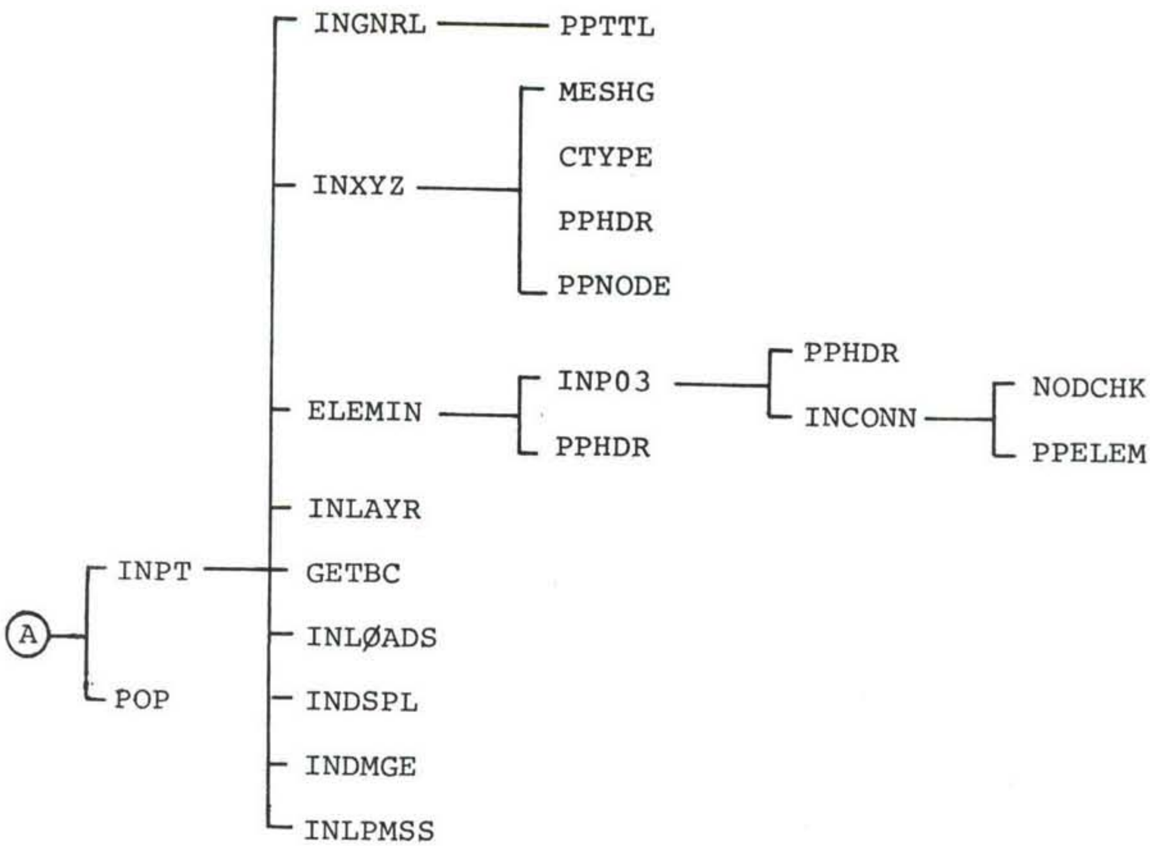
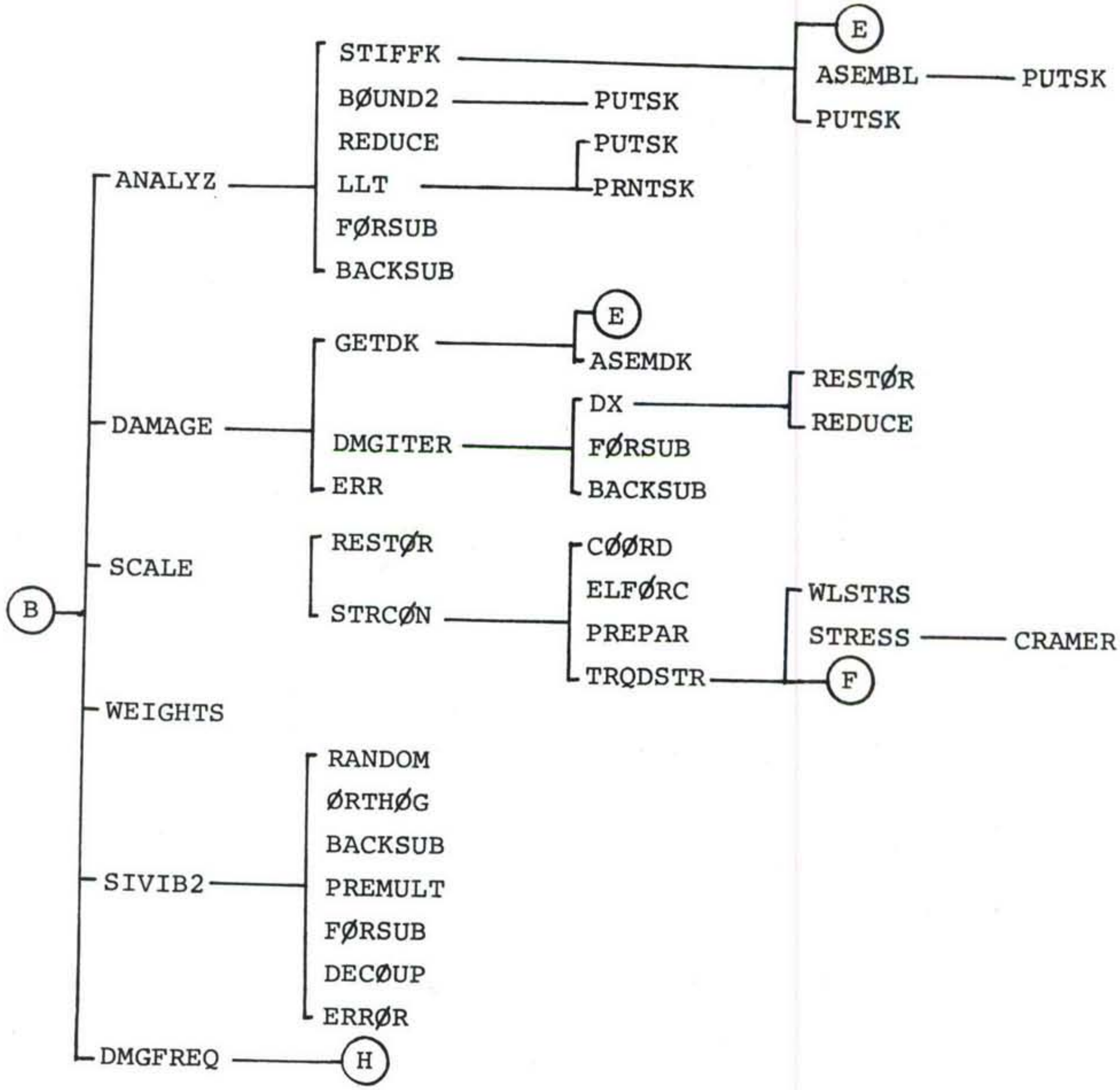
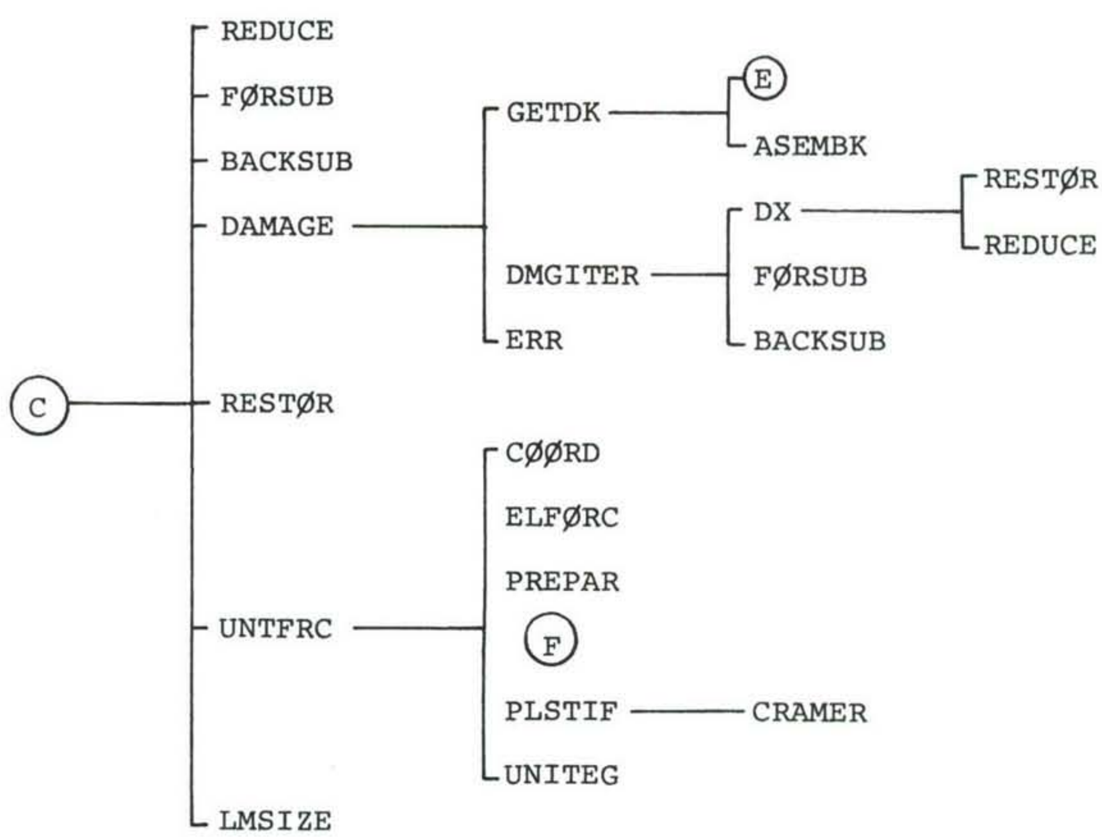


Figure 2. Input Routines: INITP Call Sequence.



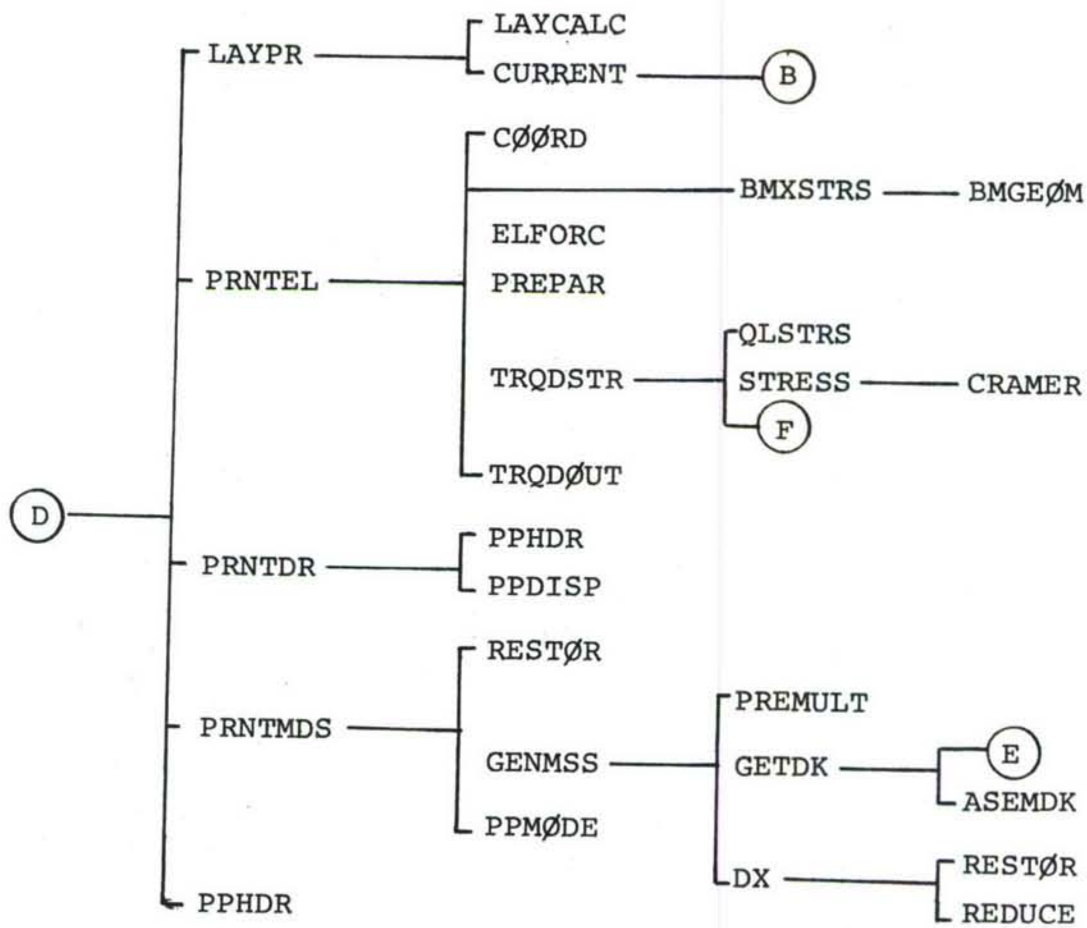
NOTE: For details of E, F, and H, see Figure 6.

Figure 3. Analysis Routines: CURRENT Call Sequence (Overall).



NOTE: For details of E and F, see Figure 6.

Figure 4. Displacement Resizing Routines: DMØDE Call Sequence.



NOTE: For details of B, see Figure 3, and for E and F, see Figure 6.

Figure 5. Print Routines: PRINT Call Sequence.

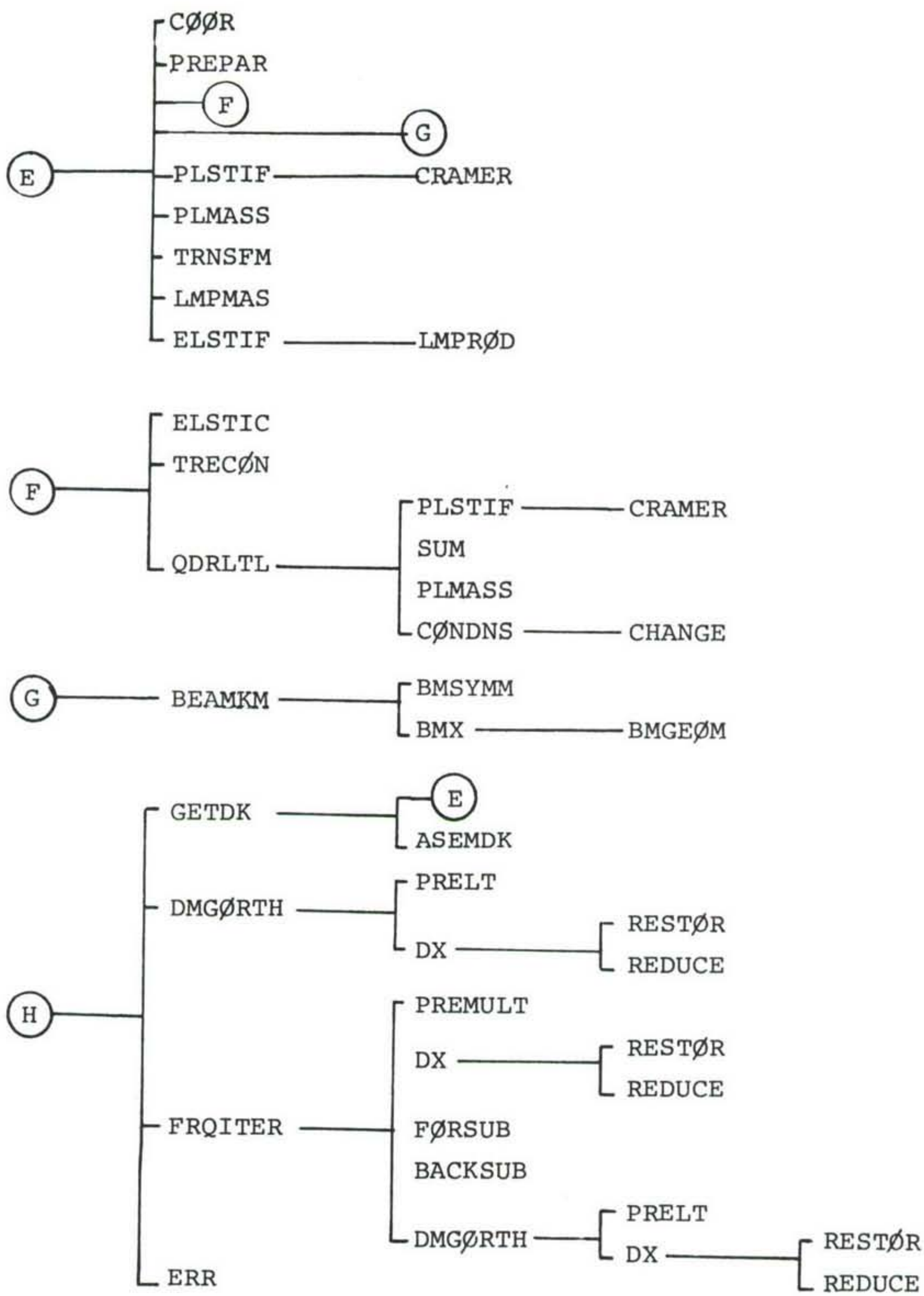


Figure 6. Analysis Routine Segments E, F, G, and H.

APPENDIX A

VARIABLE NAME / SUBROUTINE NAME CROSS-REFERENCE TABLE

UDRI - Cross reference program: XREF

This program was written using FORTRAN V on the ASD CDC Cyber 750. It is designed to read a compiler listing of a FORTRAN IV program and generate a complete list of variable and labeled common names. It also provides a set of tables which identify which variables are used and defined in which routines as well as labeled commons.

Program access:

```
ATTACH,XREF,XREFPLUS,ID=KLØØS,SN=ASDAD             (source code)
ATTACH,XX,XXPLUS,ID=KLØØS,SN=ASDAD                 (object code)
```

Compilation of source code:

```
FTN5,I=XREF,B=XX,LØ=0,ANSI=0,ØPT=2
FTN5,I=XREF,B=0,LØ=M/A/R,L=LIST,ANSI=0,ØPT=2       (for listing)
```

Preparations for execution:

TAPE1 must contain the program listing which XREF is to read. XREF considers TAPE1 to be multiple routines; all of the same executable program. This does not have to be a complete program nor arranged in any particular order. To develop a TAPE1:

FTN,I=lfn,L-TAPE1,R=3,...

TAPE1 can be editted prior to execution of XREF.

Execution:

XREF can be run at the terminal or in batch in the same manner as any other LGO file. (The presence of TAPE1 is the only requirement.) As XREF reads the symbolic reference map, it will display various error messages related to the routine it is currently processing. This is useful in finding FTN error messages in large listings. For very large listings, the OUTPUT file can be disconnected from the terminal and PAGEd after termination.

Results:

XREF will display a summary of the number of routines in TAPE1, the number of variable names in the cross-reference table, and the number of labeled commons found. It will then list all of the routine, variable, and common names in alphabetical order (listing is in 132 columns). Then XREF will report the number of 'garbage' variables and list their names. A 'garbage' variable name is one which was found to be unused in one of the routines. Some of these may occur in the variable cross-reference table, others may not. XREF stores the cross reference tables on TAPE2. This file is rewound at the end of the program and can be printed by: COPY,TAPE2,OUTPUT. TAPE2 is constructed in two parts; variables and commons. The names are listed in alphabetical order (by internal CDC display code) with the routines they are found in listed beside them. For variable names, an asterisk will appear with the routines that were found to define the variable in the R=3 map. The tables are designed to be reproduced onto a 8 1/2 X 11 inch paper with regular typing margins (and no reduction).

A :	ANALYZ	*CRAMER	DAMAGE	DMGFREQ	PLSTIF
	SIVIB2	STRESS			
AA :	*COORD	ELFORC	ELSTIF	LMPMAS	*LMPROD
	TRECON	TRNSFM			
AAE :	AVERAGE	GETBEST	LAYCALC	*LAYPR	*PRINT
	PRNTEL	*SAVE	TRQDOUT		
AAEX :	AVERAGE	GETBEST	*SAVE		
AAEY :	AVERAGE	GETBEST	*SAVE		
AAX :	*STRESS				
AAXY :	*STRESS				
AAY :	*STRESS				
AB :	*COORD				
ADR :	*SCALE	*UNITEG			
AE :	*AVERAGE	CURRENT	*DMODE	*ELSTIF	*EMODE
	*GETBEST	GETDK	*INCONN	*LAYPR	LMPMAS
	*LMPROD	*LMSIZE	PREPAR	PRINT	PRNTEL
	SAVE	SCALE	STIFFK	STRCON	*TRECON
	TRQDSTR	UNITEG	WEIGHTS		
AEMAX :	EMODE	*INCONN	SCALE		
AEMNM :	AVERAGE	DMODE	*EMODE	*INCONN	
AEMNMM :	INCONN	*INGNRL			
AEX :	*AVERAGE	*GETBEST	*INLAYR	LAYCALC	*LAYPR
	*LMSIZE	PREPAR	SAVE	TRQDOUT	
AEXMIN :	*INLAYR	LMSIZE			
AEXYMIN:	*INLAYR	LMSIZE			
AEY :	*AVERAGE	*GETBEST	*INLAYR	LAYCALC	*LAYPR

	*LMSIZE	PREPAR	SAVE	TRQDOUT	
AEYMIN :	*INLAYR	LMSIZE			
AH :	*LMPROD				
AL :	*COORD	ELSTIF	GETDK	LMPROD	PRNTEL
	STIFFK	STRCON	UNITEG		
ALS :	*PREPAR	PRNTEL	QLSTRS	STRCON	STRESS
ALSTRS :	*INP03	PREPAR			
AM :	*LMPMAS				
AMAX :	*LMSIZE	*QLSTRS	*STRCON		
AMAXAE :	*AVERAGE	*DMODE	*EMODE		
AMX :	*SCALE				
AREA :	*PLMASS	*PLSTIF	*QDRLTL		
AX :	*CHANGE	*LMSIZE	*PREPAR	STRESS	*TRECON
AXX :	*TRECON				
AY :	*LMSIZE	*PREPAR	STRESS	*TRECON	
AYY :	*TRECON				
AZ :	*LMSIZE	*PREPAR	STRESS	*TRECON	
B :	*ELSTIF	*PREMULT	*TRNSFM		
BA :	*PREPAR	*TRQDSTR			
BAE :	*PREPAR				
BASEA :	AVERAGE	DMODE	EMODE	LAYCALC	LAYPR
	LMSIZE	PRINT	PRNTEL	*SCALE	*STRCON
	WEIGHTS				

BB	:	*ELSTIF				
BND	:	*PRNTDR				
C	:	ASEMBL	ASEMDK	*ELSTIF	GETDK	STIFFK
CARD	:	*ECHO				
CC	:	ASEMBL	ASEMDK	*ELSTIF	GETDK	*LMPMAS
		*LMPROD	STIFFK			
CCC	:	*TRNSFM				
CTA	:	*TRECON				
D	:	*DAMAGE	*DMGITER	*DMGORTH	*DX	*ORTHOG
		*RESTOR				
DEFLMT	:	DMODE	*INDSPL	SCALE		
DEFMAX	:	*INDSPL				
DET	:	*CONDNS				
DINCR	:	DMODE	*INGNRL			
DISP	:	*PRNTDR				
DK	:	*ASEMDK	DMGITER	DMGORTH		
DKF	:	*INDMGE				
DKFCTR	:	ASEMDK	*INDMGE	PRNTEL	STRCON	TRQDSTR
		UNITEG				
DM	:	*ASEMDK	FRQITER	GENMSS		
DMFCTR	:	ASEMDK	*INDMGE			
DONE	:	*INCONN	*NODCHCK			
DR	:	ANALYZ	CURRENT	DMODE	*ELFORC	*INLOADS
		PPDISP	PRNTDR	PRNTEL	SCALE	STRCON

	UNTFRC	*WEIGHTS			
DRATIO :	*SCALE				
DTEXTRA:	*DAMAGE				
DTITER :	*DAMAGE				
DX :	*INXYZ				
DY :	*INXYZ				
DZ :	*INXYZ				
E :	*TRQDSTR				
E1 :	ELSTIC	ELSTIF	*PREPAR	PRNTEL	STRCON
	UNITEG				
E2 :	ELSTIC	*PREPAR			
EDDR :	*QLSTRS				
EDR :	*ELFORC	PRNTEL	*QLSTRS	STRCON	TRQDSTR
	UNITEG	UNTFRC			
EDR1 :	UNITEG	UNTFRC			
EE :	*ELSTIC	PLSTIF	STRESS	*TRECON	
EEK :	*GETDK	*PREPAR	*STIFFK	*UNTFRC	
EEKK :	*ELSTIF	*PLSTIF			
EEKM :	*SUM				
EEM :	*GETDK	*PREPAR	*STIFFK		
EEMM :	*ELSTIF	*LMPROD	*PLMASS		
EFFSTR :	*QLSTRS	TRQDSTR			
EFSTRS :	*PRNTEL	QLSTRS	*STRESS	TRQDOUT	*TRQDSTR

EK	:	*CONDNS	*GETDK	*QDRLTL	*STIFFK	UNITEG
		*UNTFRC				
EKK	:	*CONDNS	QDRLTL	QLSTRS		
EKM	:	*CHANGE	*SUM	*TRNSFM		
EL	:	*DECOUP	*DMGORTH	*ERROR	*FORSUB	*LLT
		*ORTHOG				
EL1	:	*DECOUP				
ELASM	:	*INP03				
ELCNST	:	*INP03	PREPAR			
ELEENG	:	*PRNTEL	*QLSTRS	*STRCON	TRQDOUT	*TRQDSTR
ELENG	:	*PREPAR	TRQDOUT	*TRQDSTR		
ELENG1	:	*UNITEG				
ELENTH	:	DMODE	EMODE	*GETDK	LMPMAS	LMSIZE
		*STIFFK	WEIGHTS			
ELL	:	*DECOUP				
EM	:	*CONDNS	*GETDK	*QDRLTL	*STIFFK	
EMM	:	*CONDNS	*LMPMAS	QDRLTL		
ENG	:	*STRESS				
ENGCAP	:	SCALE	*STIFFK			
ENGG	:	QLSTRS				
ENGLTA	:	*DMODE	*EMODE	*STIFFK		
ENGMAX	:	*TRQDSTR				
ENGST1	:	*DMODE	UNITEG			

ENGSTR	:	*SCALE			
ENGTOT	:	*PRNTEL	*STRCON	*TRQDSTR	
ENGX	:	LMSIZE	*TRQDSTR	*UNITEG	
ENGXY	:	LMSIZE	*TRQDSTR	*UNITEG	
ENGY	:	LMSIZE	*TRQDSTR	*UNITEG	
EONE	:	*PLSTIF			
ER	:	*ERROR			
ERR	:	*ERROR	*INCONN	*NODCHCK	
ERR0	:	*DAMAGE			
ERR1	:	*DAMAGE			
ERREST	:	*DMGFREQ			
ERRMAX	:	*ERR			
ERROR	:	*ERR			
ESRTIO	:	*PREPAR	*STRCON	TRQDOUT	*TRQDSTR
ETA	:	*COORD	CRAMER		
ETWO	:	*PLSTIF			
EX	:	*PLSTIF	*STRESS		
EXY	:	*STRESS			
EY	:	*STRESS			
F	:	*REDUCE			
FCTR	:	*PRNTEL	*STRCON	*TRQDSTR	*UNITEG

FR	:	ANALYZ	*INLOADS	PRNTDR	SCALE	
GENM	:	*GENMSS	PPMODE	PRNTMDS		
GES	:	*RANDOM				
GM	:	*ASEMBL	*BOUND2	PREMULT	*STIFFK	
I	:	*ASEMBL	*ASEMDK	*BACKSUB	*BOUND2	*CHANGE
		*CONDNS	*CRAMER	*CURRENT	*DAMAGE	*DECOUP
		*DMGFREQ	*DMGITER	*DMGORTH	*DMODE	*DX
		*ELFORC	*ELSTIC	*ELSTIF	*EMODE	*ERR
		*ERROR	*FORSUB	*GENMSS	*GETBC	*GETBEST
		*GETDK	*INCONN	*INDMGE	*INDSPL	*INITIAL
		*INLAYR	*INLOADS	*INLPMSS	*INP03	*INXYZ
		*LLT	*LMPMAS	*LMPROD	*LMSIZE	*ORTHOG
		*PLMASS	*PLSTIF	*POP	*PPDISP	*PPELEM
		*PPMODE	*PPNODE	*PRELT	*PREMULT	*PREPAR
		*PRINTK	*PRNTDR	*PRNTEL	*PRNTMDS	*QDRLTL
		*QLSTRS	*RANDOM	*REDUCE	*RESTOR	*SCALE
		*SIVIB2	*STIFFK	*STRESS	*SUM	*TRECON
		*TRNSFM	*TRQDOUT	*TRQDSTR	*UNITEG	*UNTFRC
		*WEIGHTS				
I0	:	*BACKSUB	*FORSUB	*INCONN	*POP	*PRELT
		*STIFFK				
I1	:	*BACKSUB	*BOUND2	*DECOUP	ELEMIN	*ELSTIF
		*FORSUB	*INDMGE	*INDSPL	*LMPROD	*POP
		*PPHDR	PRINT	*STIFFK		
I2	:	ELEMIN	INXYZ	*PPHDR	PRINT	*STRESS
I2M1	:	*STRESS				
I3	:	ELEMIN	INXYZ	*PPHDR	PRINT	PRNTDR
I4	:	ELEMIN	INP03	INXYZ	*PPHDR	PRINT
		PRNTDR				
IA	:	*BOUND2	*TRNSFM			
IAA	:	*ASEMBL	*ASEMDK			

IAREAS	:	INCONN	*INGNRL	INLAYR		
IBC	:	*GETBC				
IBND	:	BOUND2	*GETBC	PRNTDR	REDUCE	RESTOR
ICOL	:	BACKSUB	*BOUND2	FORSUB	LLT	*POP
		PRELT	PREMULT			
ICTYPE	:	*INXYZ				
IDIAG	:	ASEMBL	BACKSUB	*BOUND2	FORSUB	LLT
		*POP	PRELT	PREMULT	PRINTK	STIFFK
IDKMK	:	*PRELT				
IDMG	:	GETDK	*INDMGE			
IELNO	:	*INCONN	*NODCHCK			
IELPR	:	*INCONN				
IELPRM1	:	*INCONN				
IFLAG	:	*RANDOM				
IH	:	*BOUND2	*REDUCE	*RESTOR	*SUM	
IHH	:	*SUM				
II	:	*DECOUP	*FORSUB	*GETDK	*PLSTIF	*PREMULT
		*STIFFK	*TRQDSTR	*UNITEG	*UNTFRC	
IJ	:	*ASEMDK	DX			
IL	:	*INLOADS	*PPDISP			
IL1	:	*DECOUP				
IM	:	*INDSPL	*INLOADS			
IM1	:	*BACKSUB	*POP			

```
IMAX   :  DAMAGE   DMGFREQ  *ERR

IMODN  : *UNITEG

INCR   : *GETBC

IND    : *TRECON

INDANG : *INGNRL    INLAYR    TRECON

INDEX  : *PREPAR
INDMIN :  INCONN   *INGNRL

INDX   : *UNITEG

INT    : *SIVIB2

IP     : *ELSTIC

IP1    : *DMGFREQ  *FRQITER

IPP    : *INGNRL    PPDISP    PPELEM    PPHDR     PPNODE
          PPTTL     PRNTMDS

IPR    : *INCONN

IQ     : *BACKSUB  *FORSUB   *LLT      *PREMULT

ISOTRN : *ELEMIN    INP03     PREPAR

ISYS   : *INXYZ

IT     : *BACKSUB  *DAMAGE   *DMGFREQ

ITRI   : *POP

ITYPE  : *CTYPE    *ELEMIN   *INCONN   *INXYZ    *NODCHCK
         *PPHDR    *PRNTEL    TRQDOUT

IX     : *CHANGE   *PLSTIF

IY     : *CHANGE
```

J	:	*ASEMBL	*ASEMDK	*BACKSUB	*BOUND2	*CHANGE
		*CONDNS	*CRAMER	*DAMAGE	*DECOUP	*DMGFREQ
		*DMGITER	*DMGORTH	*DMODE	*DX	*ELFORC
		*ELSTIC	*ELSTIF	*ERR	*ERROR	*FORSUB
		*FRQITER	*GENMSS	*GETBC	*GETDK	*INCONN
		*INDMGE	*INDSPL	*INLOADS	*INP03	*INXYZ
		*LLT	*LMPMAS	*LMPROD	*ORTHOG	*PLMASS
		*PLSTIF	*POP	*PPDISP	*PPMODE	*PRELT
		*PREMULT	*PREPAR	*PRNTDR	*PRNTEL	*PRNTMDS
		*QDRLTL	*QLSTRS	*RANDOM	*REDUCE	*RESTOR
		*SCALE	*SIVIB2	*STIFFK	*STRCON	*SUM
		*TRNSFM	*TRQDSTR	*UNITEG	*UNTFRC	*WEIGHTS
J0	:	*INDSPL	*TRQDSTR			
J1	:	*DMODE	*ELSTIF	*INCONN	*INDSPL	*INXYZ
		*LMPROD	*PRNTEL	*STRCON	*TRQDSTR	*UNITEG
J3	:	*PRNTMDS				
J3M2	:	*PRNTMDS				
JA	:	*BOUND2	*TRNSFM			
JAA	:	*ASEMDK	*TRNSFM			
JD	:	*GETBC				
JH	:	*INP03	*SUM			
JJ	:	*INXYZ				
JJ1	:	*LLT				
JL	:	*INP03				
JL1	:	*DECOUP	*ERROR			
JM	:	*INDSPL	*INLOADS			
JNN	:	*PRNTDR				

JX	:	*ASEMBL	*BOUND2			
K	:	*BACKSUB	*BOUND2	*CONDNS	*DECOUP	*DMGFREQ
		*DMGORTH	*DX	*ELFORC	*FORSUB	*FRQITER
		*GENMSS	*INDSPL	*INLOADS	*LLT	*LMPMAS
		*LMPROD	*ORTHOG	*PLSTIF	*PPMODE	*PRELT
		*PREMULT	*PRINTK	*PRNTDR	*PRNTEL	*PRNTMDS
		*QLSTRS	*RANDOM	*REDUCE	*RESTOR	*SCALE
		*STRCON	*STRESS	*TRNSFM	*TRQDOUT	*TRQDSTR
		*UNITEG				
KO	:	*PRNTEL	*STRCON	*UNITEG		
K1	:	*GETDK	*LMPMAS	*PRNTEL	*STRCON	*TRQDSTR
		*UNITEG				
K2	:	*LMPMAS				
KA	:	*TRNSFM				
KAA	:	*TRNSFM				
KANLYZE	:	DOPROB	*INGNRL	LAYPR	SCALE	
KDEFEQ	:	*DMODE	UNITEG			
KDMG	:	*ASEMDK	*GETDK			
KF	:	*UNITEG				
KGEN	:	*INCONN				
KH	:	*ASEMBL	*ASEMDK	*BOUND2	*ELFORC	*EMODE
		*INDSPL	*INLOADS	*INPO3	*PPDISP	*PRNTDR
		*UNITEG				
KHH	:	*ASEMBL	*ASEMDK	*PPDISP	*PRNTDR	
KII	:	*ASEMDK				
KK	:	*ELFORC	*LLT	*PREPAR	*TRQDOUT	
KL	:	*PRNTEL				

KL1 :	*DECOUP	*PREMULT			
KSAVE :	DOPROB	*WEIGHTS			
KSTR :	*DAMOPT	WEIGHTS			
KTR :	*QLSTRS	TRQDOUT			
KX :	*ASEMBL	*ASEMDK	*BOUND2	*CHANGE	*CONDNS
	*DMODE	*ELFORC	GETDK	*INDSPL	*INLOADS
	*LMPMAS	*POP	*PREPAR	*PRINTK	*QLSTRS
	STIFFK	*SUM	TRQDOUT	TRQDSTR	*UNITEG
	UNTFRC				
KXX :	*BOUND2				
KY :	*ASEMBL	*ASEMDK	*BOUND2	*CHANGE	*CONDNS
	GETDK	*INLOADS	*LMPMAS	*POP	*PREPAR
	*PRINTK	*QLSTRS	STIFFK	*SUM	*UNITEG
	UNTFRC				
L :	*ASEMBL	*ASEMDK	*AVERAGE	*COORD	*DAMAGE
	*DMGITER	*DMODE	*DX	*ELFORC	*EMODE
	*ERR	*GETDK	*LAYCALC	*LAYPR	*LMPMAS
	*LMSIZE	*PLSTIF	*POP	*PREMULT	*PREPAR
	*PRINT	*PRNTEL	*PRNTMDS	*QDRLTL	*QLSTRS
	*REDUCE	*RESTOR	*SAVE	*SCALE	*STIFFK
	*STRCON	*TRECON	*TRQDOUT	*TRQDSTR	*UNITEG
	*UNTFRC	*WEIGHTS			
L1 :	DECOUP	ERROR	*ORTHOG	*PRNTMDS	RANDOM
	*SIVIB2				
LAM :	AVERAGE	GETBEST	*INCONN	INLAYR	LAYCALC
	LAYPR	LMSIZE	PREPAR	TRQDOUT	*TRQDSTR
LAMFO :	*INCONN				
LAYERD :	DMODE	EMODE	*INGNRL	INPT	PRINT
	SAVE				
LDELTA :	*PRNTMDS				

LDS :	*ELFORC	*PREPAR			
LET :	*ERROR	PRNTMDS	*SIVIB2		
LFLAG :	*LAYCALC	LAYPR			
LINES :	*INLAYR	*LAYPR	*PRNTDR	*PRNTEL	
LINNUM :	*ECHO				
LM :	*LAYCALC				
LMNODE :	*INLPMSS	STIFFK			
LMTDSP :	INDSPL	*INGNRL	INPT	SCALE	
LMTSTR :	*INGNRL	SCALE			
LOADS :	ANALYZ	CURRENT	GETDK	*INLOADS	INPT
	PRNTEL	STIFFK	STRCON	TRQDSTR	UNITEG
	WEIGHTS				
LOCK :	*ERROR	*ORTHOG	*SIVIB2		
LT :	*LAYCALC				
M :	*ASEMDK	*DAMAGE	*GETDK	*PRNTMDS	*UNITEG
M1 :	*BACKSUB	*FORSUB	*LLT	*PREMULT	*UNITEG
M2 :	*ASEMBL	*ASEMDK	*TRNSFM		
M3 :	*TRNSFM				
MA :	ASEMBL	ASEMDK	COORD	*CRAMER	ELFORC
	*INCONN	*PLSTIF	POP	PPELEM	PRNTEL
	*STRESS	*SUM	TRQDOUT		
MAA :	*INITIAL	QDRLTL	QLSTRS		
MAXDCCL:	DOPROB	*INGNRL			

MAXDDIT:	DAMAGE	*INDMGE			
MAXDFIT:	DMGFREQ	*INDMGE			
MAXECCL:	DOPROB	*INGNRL			
MAXSK :	*INITIAL	INITP			
MAXSZE :	EMODE	*INGNRL	SCALE		
MB :	ASEMBL	ASEMDK	*CONDNS	COORD	*CRAMER
	ELFORC	*INCONN	LMPMAS	*PLSTIF	POP
	PPELEM	PRNTEL	QDRLTL	*STRESS	*SUM
	TRQDOUT				
MBB :	*INITIAL	QDRLTL	QLSTRS		
MC :	ASEMBL	ASEMDK	*CONDNS	COORD	*CRAMER
	ELFORC	*INCONN	LMPMAS	*PLSTIF	POP
	PPELEM	QDRLTL	*STRESS	*SUM	TRQDOUT
MCC :	*INITIAL	QDRLTL	QLSTRS		
MD :	ASEMBL	ASEMDK	*CONDNS	COORD	ELFORC
	*INCONN	LMPMAS	POP	PPELEM	QDRLTL
	TRQDOUT				
MDEFEQ :	*DMODE	UNITEG			
MEMBS :	AVERAGE	CURRENT	DMODE	*ELEMIN	EMODE
	GETBEST	INCONN	INDMGE	INLAYR	INP03
	LAYPR	LMSIZE	POP	PPELEM	PRINT
	PRNTEL	SAVE	SCALE	STIFFK	STRCON
	UNTFRC	WEIGHTS			
MESS :	*NODCHCK				
MGEN :	*INXYZ				
ML :	*DAMAGE				
MLPI :	*DAMAGE				

MLPJ	:	*DAMAGE				
MM	:	ASEMBL	ASEMDK	ELFORC	ELSTIF	GETBC
		INDSPL	*INGNRL	INLOADS	INPT	LMPROD
		POP	PPDISP	PPMODE	PRNTDR	STIFFK
		TRNSFM				
MM2	:	*PPMODE				
MNLAYR	:	*INGNRL	INLAYR			
MODES	:	ASEMBL	ASEMDK	BOUND2	CONDNS	CURRENT
		DMGFREQ	DMGORTH	ELSTIF	ERROR	FRQITER
		GENMSS	GETDK	*INGNRL	PPMODE	PREPAR
		PRINT	PRNTMDS	QDRLTL	SIVIB2	STIFFK
MSK	:	*DX				
MXMEMB	:	*SCALE				
MYOUNG	:	DMODE	EMODE	*INCONN	INLAYR	PREPAR
		PRNTEL	WEIGHTS			
N	:	*DX	*INXYZ	*PRELT	*PREMULT	
N1	:	*GETBC	*INXYZ	*LLT		
N2	:	*GETBC				
N3	:	*GETBC				
NA	:	*ASEMBL	*ASEMDK	*SUM		
NAA	:	*ASEMBL				
NACTIVE	:	DMODE	*INITIAL	SCALE		
NBC1	:	*GETBC				
NBC2	:	*GETBC				
NBNDRY	:	BOUND2	*GETBC	INPT	PRNTDR	REDUCE
		RESTOR				

NCASES :	DMODE	*INPT	PRNTDR	PRNTEL	QLSTRS
	SCALE	STRCON	STRESS	TRQDOUT	TRQDSTR
	UNTFRC	WEIGHTS			
NCOMP :	*ELEMIN				
NCON :	*ELFORC				
ND :	*GETBC				
NDCASES:	*DMODE	UNITEG	UNTFRC		
NDCYCL :	*DOPROB	*INITP	WEIGHTS		
NDEFEQ :	*SCALE				
NDK :	*ASEMDK	DX	*GETDK		
NDMG :	GETDK	*INDMGE			
NDMGCAS:	CURRENT	DAMAGE	DMGFREQ	DMODE	GENMSS
	INDMGE	*INGNRL	INLOADS	INPT	PPMODE
	PRNTEL	PRNTMDS	STRCON	TRQDSTR	UNITEG
	WEIGHTS				
NDSP :	*ELFORC				
NDUMMY :	*DMODE	DOPROB	UNITEG		
NECYCL :	*DOPROB	*INITP	WEIGHTS		
NELEM :	*ELEMIN				
NFDEG :	*LAYCALC	LAYPR			
NFIL :	*INITIAL	PPDISP	PPELEM	PPHDR	PPNODE
	PPTTL				
NH :	*BOUND2	*REDUCE	*RESTOR		
NH1 :	*REDUCE				

NINCR	:	*INXYZ				
NJLODS	:	*INLOADS				
NL1	:	*ORTHOG	*RANDOM			
NLAM	:	*LAYPR				
NLINES	:	*ECHO				
NLMPMSS	:	*INGNRL	INLPMSS	INPT	STIFFK	WEIGHTS
NM	:	BACKSUB	DAMAGE	DECOUP	DMGFREQ	DMGITER
		DMGORTH	ERR	ERROR	FORSUB	FRQITER
		GENMSS	*INPT	LLT	ORTHOG	PRELT
		PREMULT	PRINTK	PRNTMDS	RANDOM	SCALE
		SIVIB2				
NMAT	:	*ELEMIN	INP03			
NMT	:	*ELEMIN				
NN	:	BOUND2	DMODE	DX	INDSPL	INLOADS
		*INPT	POP	PPMODE	REDUCE	RESTOR
		SCALE	WEIGHTS			
NND	:	*ASEMBL	*ASEMDK	*ELFORC	*LMPMAS	*STRESS
		*TRNSFM				
NNDEG	:	*LAYCALC	LAYPR			
NNODES	:	ASEMBL	ASEMDK	COORD	DMODE	ELFORC
		EMODE	GETDK	*INCONN	INLAYR	LMPMAS
		POP	PREPAR	PRNTEL	QDRLTL	QLSTRS
		STIFFK	STRCON	*TRNSFM	TRQDOUT	TRQDSTR
		UNITEG	UNTFRC	WEIGHTS		
NNRM	:	*QDRLTL				
NO	:	*CONDNS	*QDRLTL			
NOD	:	*INCONN	*NODCHCK			

NODES	:	INDSPL	INPT	*INXYZ	PPDISP	PPMODE
		PPNODE	PRNTDR	PRNTMDS		
NOI	:	*INGNRL	SIVIB2			
NONORM	:	*PLSTIF				
NONZRO	:	INITP	*POP	STIFFK		
NP	:	*INXYZ				
NP1	:	*GENMSS	*PPMODE	*PRNTMDS		
NPAGE	:	*AVERAGE	*DAMAGE	*DMGFREQ	*DMODE	*DOPROB
		*EMODE	*GETBC	*GETBEST	*INCONN	*INDMGE
		*INDSPL	*INGNRL	*INITP	*INLAYR	*INLOADS
		*INLPMSS	*INPO3	*INXYZ	*LAYPR	*NODCHCK
		*POP	*PRINTK	*PRNTDR	*PRNTEL	*PRNTMDS
NSTR	:	DAMOPT	*INITIAL			
NTRIAL	:	DECOUP	ERROR	*ORTHOG	RANDOM	*SIVIB2
NUFR	:	*DMODE				
NWORK	:	INXYZ				
NX	:	*DMODE				
NY	:	*RANDOM				
NZDEG	:	*LAYCALC	LAYPR			
PERMBA	:	*SCALE				
PMU	:	ELSTIC	*PREPAR			
PMU1	:	*ELSTIC				
POISON	:	*INPO3	PREPAR			
Q	:	*DECOUP				

QUAD	:	TRQDOUT	TRQDSTR	UNTFRC		
RO	:	*DAMAGE				
R1	:	*DAMAGE	ELEMIN	INPO3	INXYZ	*PPHDR
		PRINT	PRNTDR			
RAD	:	*INLAYR				
RHO	:	ELSTIF	LMPMAS	LMPROD	PLMASS	*PREPAR
RHO1	:	DMODE	EMODE	*INPO3	PREPAR	WEIGHTS
RHOA	:	*PLMASS				
S	:	*ERR	*INLOADS	*UNITEG		
SAVE	:	*CONDNS	*LMPMAS	*TRECON		
SHEARM	:	*INPO3				
SK	:	*ASEMBL	BACKSUB	*BOUND2	FORSUB	*LLT
		PRELT	*STIFFK			
SKGM	:	*PRINTK				
SM	:	ELSTIC	*PREPAR			
SPRDF	:	DMODE	EMODE	*INGNRL	STIFFK	
SQRTI2	:	*TRECON				
SSX	:	*QLSTRS	TRQDOUT			
SSXY	:	*QLSTRS	TRQDOUT			
SSY	:	*QLSTRS	TRQDOUT			
STA	:	*TRECON				
STRENG	:	DMODE	*EMODE	*STRCON	*TRQDSTR	*UNITEG
STRMAX	:	*SCALE				

SX	:	*PRNTEL	QLSTRS	*STRCON	*STRESS	TRQDOUT
SXY	:	QLSTRS	*STRESS	TRQDOUT		
SY	:	QLSTRS	*STRESS	TRQDOUT		
T	:	*DMGORTH	*ERR			
TDR1	:	*RESTOR				
TDR2	:	*RESTOR				
TEMP	:	*INDSPL	*INLOADS	*LMSIZE	*NODCHCK	
TEXTRA	:	*DAMAGE				
TFFR	:	GETDK	*LAYCALC	*PREPAR	STIFFK	TRQDOUT
		TRQDSTR	UNTFRC			
TFR	:	*PREPAR	PRNTEL	TRQDOUT		
THCKMN	:	*INCONN				
THCKMX	:	*INCONN				
THICK	:	*INCONN				
THKLAM	:	*INGNRL	INLAYR	LAYCALC	LAYPR	TRQDSTR
TIME0	:	*ANALYZ	*DAMAGE	*DMGFREQ	*SIVIB2	
TIME1	:	*ANALYZ	*DAMAGE	*DMGFREQ	*SIVIB2	
TIME2	:	*ANALYZ	*DAMAGE			
TIME3	:	*ANALYZ				
TITER	:	*DAMAGE				
TITLE	:	AVERAGE	DAMAGE	DMGFREQ	DMODE	DOPROB
		EMODE	GETBC	GETBEST	INCONN	INDMGE
		INDSPL	*INGNRL	INLAYR	INLOADS	INLPMSS

	INPO3	INXYZ	LAYPR	NODCHCK	POP
	PPMODE	PPTTL	PRINTK	PRNTDR	PRNTEL
	PRNTMDS				
TOLDMGD:	DAMAGE	*INDMGE			
TOLDMGF:	DMGFREQ	*INDMGE			
TOLVEC :	DECOUP	ERROR	*INGNRL		
TRANG :	LMPMAS	*QDRLTL	TRQDSTR		
TRIANG :	*CRAMER	QDRLTL	STRESS	TRQDOUT	UNTFRC
TTHK :	*GETDK	PLMASS	PLSTIF	*STIFFK	*TRQDSTR
	*UNTFRC				
TWOPII :	*PRNTMDS				
U :	*BACKSUB	*DAMAGE	*DECOUP	*DMGORTH	ERROR
	FRQITER	GENMSS	*PLSTIF	*SIVIB2	
UDR :	*DMODE	UNTFRC			
UV :	*STRESS				
V :	*DAMAGE	*DECOUP	DMGFREQ	*FORSUB	*FRQITER
	GENMSS	SIVIB2			
W :	*BACKSUB	*DECOUP	*DMGFREQ	ERROR	FRQITER
	GENMSS	*ORTHOG	PPMODE	*PRNTMDS	*RANDOM
	SIVIB2				
WBAR :	*WEIGHTS				
WEIGHT :	DOPROB	SAVE	*WEIGHTS		
WLMPMSS:	*INLPMSS	STIFFK	WEIGHTS		
WMAX :	*PRNTMDS				
WMEMB :	*WEIGHTS				

WSHEAR	:	*WEIGHTS
WTLAST	:	DOPROB *INITP *SAVE WEIGHTS
WW	:	*WEIGHTS
X	:	COORD *DMGITER *FORSUB INLOADS *INXYZ PPMODE PPNODE *PRELT PRNTDR PRNTMDS
X1	:	*CTYPE *INXYZ
X2	:	*CTYPE *INXYZ
X3	:	*CTYPE *INXYZ
XA	:	*INLAYR
XANG	:	*INLAYR TRECON
XCOMP	:	*COORD
XI	:	*COORD CRAMER
XL	:	*DECOUP DMGFREQ ERROR *FRQITER PPMODE *PRNTMDS *SIVIB2
XP	:	*INXYZ
Y	:	*DMGITER *DMGORTH *DX FRQITER *PRELT *PREMULT *RANDOM
Y1	:	*CTYPE
Y2	:	*CTYPE
Y3	:	*CTYPE
YA	:	*INLAYR
YANG	:	*INLAYR TRECON
YCOMP	:	*COORD

YOUNGM	:	*INP03	PREPAR
YP	:	*INXYZ	
Z	:	*DX	*PREMULT
ZA	:	*INLAYR	
ZANG	:	*INLAYR	TRECON
ZCOMP	:	*COORD	
ZP	:	*INXYZ	

APPENDIX B

COMMON BLOCK / SUBROUTINE NAME CROSS-REFERENCE TABLE

ALSTRS :	PREPAR	PRNTEL	QLSTRS	STRCON	STRESS
	TRQDOUT				
AMAXMIN:	AVERAGE	DMODE	EMODE	INCONN	INGNRL
	INLAYR	SCALE			
ANG :	INGNRL	INLAYR	TRECON	UNTFRC	
AREA :	COORD	ELSTIF	GETDK	LAYCALC	LMPMAS
	LMPROD	PLMASS	PLSTIF	PREPAR	PRNTEL
	QDRLTL	STIFFK	STRCON	STRESS	TRQDOUT
	TRQDSTR	UNITEG	UNTFRC		
BASEA :	AVERAGE	CURRENT	DMODE	ELEMIN	EMODE
	GETBEST	INCONN	INDMGE	INLAYR	INPO3
	LAYCALC	LAYPR	LMSIZE	POP	PPELEM
	PRINT	PRNTEL	SAVE	SCALE	STIFFK
	STRCON	UNTFRC	WEIGHTS		
BND :	BOUND2	GETBC	INPT	PRNTDR	REDUCE
	RESTOR				
CONN :	ASEMBL	ASEMDK	COORD	ELFORC	INCONN
	LMPMAS	POP	PPELEM	PRNTEL	QDRLTL
	TRQDOUT				
DEFLMT :	DMODE	INDSPL	INGNRL	SCALE	
DK :	ASEMDK	DMGITER	DMGORTH	DX	FRQITER
	GENMSS	GETDK			
DMG :	ASEMDK	CURRENT	DAMAGE	DMGFREQ	DMODE
	GENMSS	GETDK	INDMGE	INGNRL	INLOADS
	INPT	PPMODE	PRNTEL	PRNTMDS	STRCON
	TRQDSTR	UNITEG	WEIGHTS		
DMGERR :	DAMAGE	DMGFREQ	INDMGE		
DUMMY :	DMODE	UNITEG	UNTFRC		
EIGN :	ASEMBL	ASEMDK	BOUND2	CONDNS	CURRENT
	DECOUP	DMGFREQ	DMGORTH	ELSTIF	ERROR
	FRQITER	GENMSS	GETDK	INGNRL	PPMODE

	PREPAR	PRINT	PRNTMDS	QDRLTL	RANDOM
	SIVIB2	STIFFK			
EIGVEC :	DAMAGE	DECOUP	DMGFREQ	DMGORTH	ERROR
	FRQITER	GENMSS	PPMODE	PRNTMDS	SIVIB2
ELEM :	ASEMBL	ASEMDK	AVERAGE	COORD	CURRENT
	DMODE	ELFORC	EMODE	GETBEST	GETDK
	INCONN	INLAYR	LAYCALC	LAYPR	LMPMAS
	LMSIZE	POP	PREPAR	PRINT	PRNTEL
	QDRLTL	QLSTRS	SAVE	SCALE	STIFFK
	STRCON	TRQDOUT	TRQDSTR	UNITEG	UNTFRC
	WEIGHTS				
ENERGY :	PREPAR	PRNTEL	QLSTRS	STRCON	TRQDOUT
	TRQDSTR				
ESTRESS:	PREPAR	PRNTEL	QLSTRS	STRCON	STRESS
	TRQDOUT	TRQDSTR			
FRDR :	ANALYZ	CURRENT	DMODE	GETDK	INLOADS
	INPT	PPDISP	PRNTDR	PRNTEL	QLSTRS
	SCALE	STIFFK	STRCON	STRESS	TRQDOUT
	TRQDSTR	UNITEG	UNTFRC	WEIGHTS	
GENM :	GENMSS	PPMODE	PRNTMDS		
GM :	ASEMBL	BOUND2	PREMULT	STIFFK	
LAYENG :	LMSIZE	TRQDSTR	UNITEG		
LAYERD :	AVERAGE	DMODE	EMODE	GETBEST	INGNRL
	INLAYR	INPT	LAYCALC	LAYPR	LMSIZE
	PREPAR	PRINT	PRNTEL	SAVE	TRQDOUT
LAYMIN :	INGNRL	INLAYR	LAYCALC	LAYPR	LMSIZE
	TRQDSTR				
LAYPRNT:	LAYCALC	LAYPR			
LMASS :	INGNRL	INLPMSS	INPT	STIFFK	WEIGHTS
LMTEXCD:	DMODE	DOPROB	INITIAL	SCALE	UNITEG

	UNTFRC				
LOCAL :	COORD	CRAMER	ELFORC	ELSTIF	LMPMAS
	LMPROD	TRECON	TRNSFM		
MASS :	ASEMBL	ASEMDK	CONDNS	ELSTIF	GETDK
	LMPMAS	LMPROD	PREPAR	QDRLTL	STIFFK
MAT :	DMODE	EMODE	INPO3	PREPAR	WEIGHTS
MATAXIS:	ELSTIC	PLSTIF	PREPAR	STRESS	TRECON
N :	ASEMBL	ASEMDK	BACKSUB	BOUND2	DAMAGE
	DECOUP	DMGFREQ	DMGITER	DMGORTH	DMODE
	DX	ELFORC	ELSTIF	ERR	ERROR
	FORSUB	FRQITER	GENMSS	GETBC	INDSPL
	INGNRL	INLOADS	INPT	LLT	LMPROD
	ORTHOG	POP	PPDISP	PPMODE	PRELT
	PREMULT	PRINTK	PRNTDR	PRNTMDS	RANDOM
	REDUCE	RESTOR	SCALE	SIVIB2	STIFFK
	TRNSFM	WEIGHTS			
NMAT :	ELEMIN	INPO3	PREPAR		
NODES :	COORD	DMGFREQ	INDSPL	INLOADS	INPT
	INXYZ	MESHG	PPDISP	PPMODE	PPNODE
	PRNTDR	PRNTMDS			
NPROB :	DAMOPT	INITIAL	INITP	WEIGHTS	
OPT :	DOPROB	INDSPL	INGNRL	INITP	INPT
	LAYPR	SCALE	WEIGHTS		
POSTPR :	INGNRL	INITIAL	PPDISP	PPELEM	PPHDR
	PPNODE	PPTTL	PRNTMDS		
QUAD :	INITIAL	QDRLTL	QLSTRS		
SAVE :	AVERAGE	GETBEST	LAYCALC	LAYPR	PRINT
	PRNTEL	SAVE	TRQDOUT		
SK :	ASEMBL	BACKSUB	BOUND2	FORSUB	LLT
	PRELT	STIFFK			

SKGM	:	ASEMBL	BACKSUB	BOUND2	FORSUB	INITP
		LLT	POP	PRELT	PREMULT	PRINTK
		STIFFK				
STIFF	:	ASEMBL	ASEMDK	CONDNS	ELSTIF	GETDK
		PREPAR	QDRLTL	QLSTRS	STIFFK	UNITEG
		UNTFRC				
STRESS	:	PRNTEL	QLSTRS	STRCON	STRESS	TRQDOUT
STRNENG	:	DMODE	EMODE	INGNRL	SCALE	STIFFK
		STRCON	TRQDSTR	UNITEG		
TEMP	:	ELSTIC	ELSTIF	LMPMAS	LMPROD	PLMASS
		PREPAR	PRNTEL	STRCON	TRQDOUT	UNITEG
TITLE	:	AVERAGE	DAMAGE	DMGFREQ	DMODE	DOPROB
		EMODE	GETBC	GETBEST	INCONN	INDMGE
		INDSPL	INGNRL	INITP	INLAYR	INLOADS
		INLPMSS	INPO3	INXYZ	LAYPR	NODCHCK
		POP	PPMODE	PPTTL	PRINTK	PRNTDR
		PRNTEL	PRNTMDS			
WEIGHT	:	DOPROB	INITP	SAVE	WEIGHTS	

APPENDIX C
SUBROUTINE DESCRIPTIONS

ROUTINE NAME - Main Program

PURPOSE - Initiate Calculations

CALL SEQUENCE - N/A

ARGUMENTS - N/A

CALLED BY - N/A

EXTERNALS - INITIAL, DØPRØB

COMMON BLOCKS - NPRØB

FILE NAMES - DPØST, ØUTPUT, TAPE5, TAPE6, TAPE8, TAPE99

NOTES - The call to INITIAL sets certain constants. The DØPRØB call is in a loop which allows several problems to be run at once. It is usually more convenient to keep the data sets separate and run only one problem at a time.

ROUTINE NAME	-	ANALYZ
PURPOSE	-	Perform Static Analysis of Undamaged Design
CALL SEQUENCE	-	CALL ANALYZ
ARGUMENTS	-	None
CALLED BY	-	CURRENT
EXTERNALS	-	BACKSUB, BOUND2, FORSUB, LLT, REDUCE, SECOND, STIFFK
COMMON BLOCKS	-	FRDR
FILE NAMES	-	TAPE6
NOTES	-	Calls all the routines which form the mass and stiffness matrices, imposes the boundary conditions, decomposes the stiffness matrix, and solves for the displacements.

ROUTINE NAME	-	ASEMBL
PURPOSE	-	Assemble the Global Mass and Stiffness Matrices
CALL SEQUENCE	-	CALL ASEMBL(L)
ARGUMENTS	-	L = Element Number

CALLED BY	-	STIFFK
EXTERNALS	-	PUTSK, SK
COMMON BLOCKS	-	CØNN, EIGN, ELEM, GM, MASS, N, SKGM, STIFF
FILE NAMES	-	None

NOTES	-	None

ROUTINE NAME - ASEMDK

PURPOSE - Assemble the Damaged Mass and Stiffness Matrices

CALL SEQUENCE - CALL ASEMDK(L,M,KDMG)

ARGUMENTS - L = Damaged Element Number

M = Damage Case

KDMG = Index of Damaged Element in DMFCTR Array

CALLED BY - GETDK

EXTERNALS - None

COMMON BLOCKS - CØNN, DK, DMG, EIGN, ELEM, MASS, N, STIFF

FILE NAMES - None

NOTES - None

ROUTINE NAME	-	AVERAGE
PURPOSE	-	Average Current Design with Least Weight Design
CALL SEQUENCE	-	CALL AVERAGE
ARGUMENTS	-	None
CALLED BY	-	DØPRØB
EXTERNALS	-	None
COMMON BLOCKS	-	AMAXMIN, BASEA, ELEM, LAYERD, SAVE, TITLE
FILE NAMES	-	TAPE6
NOTES	-	None

ROUTINE NAME	-	BACKSUB
PURPOSE	-	Backsubstitution
CALL SEQUENCE	-	CALL BACKSUB(W,U,I0,I1)
ARGUMENTS	-	W = Solution Vector
		U = Right Hand Side
		I0 = First Vector to be Used
		I1 = Last Vector to be Used

CALLED BY	-	ANALYZ, DMGITER, DMØDE, FRQITER, SIVIB2
EXTERNALS	-	None
COMMON BLOCKS	-	N, SKBM
FILE NAMES	-	none

NOTES	-	$L^TW = U$ is solved. The W and U need not be in separate locations.

ROUTINE NAME - BEAMKM

PURPOSE - Compute Three-dimensional Beam Element Mass and Stiffness

CALL SEQUENCE - CALL BEAMKM(ITYPE,ITRAN,EM,CC,XL)

ARGUMENTS - ITYPE = 0 For Stiffness Calculations
> 0 For Mass Calculations

ITRAN = 0 For 3-D Beam
= 1 For Plane Beam

EM = Element Matrix

CC = Axial Strains

XL = Length of Beam

CALLED BY - GETDK, STIFFK

EXTERNALS - BMSYMM, BMX

COMMON BLOCKS - BMPRØB, TEMP

FILE NAMES - None

NOTES - See Vol. I for discussion of beam element.

ROUTINE NAME	-	BMGEØM
PURPOSE	-	Geometry Calculation for Beam
CALL SEQUENCE	-	CALL BMGEØM(TR)
ARGUMENTS	-	TR = Geometry Matrix
CALLED BY	-	BMX
EXTERNALS	-	None
COMMON BLOCKS	-	NV
FILE NAMES	-	None
NOTES	-	None

ROUTINE NAME - BMSYMM

PURPOSE - Make the Beam Element Matrices Symmetric

CALL SEQUENCE - CALL BMSYMM(EM,IØRDER)

ARGUMENTS - EM = Mass or Stiffness Matrix

IØRDER = Array Containing DOF Ordering

CALLED BY - BEAMKM

EXTERNALS - None

COMMON BLOCKS - None

FILE NAMES - None

NOTES - None

ROUTINE NAME - BMX

PURPOSE - Sort the Beam DOF's

CALL SEQUENCE - CALL BMX(EM,IØRDER)

ARGUMENTS - EM = Mass or Stiffness Matrix

IØRDER = Array Containing DOF Ordering

CALLED BY - BEAMKM

EXTERNALS - BMGEØM

COMMON BLOCKS - None

NOTES - None

ROUTINE NAME - BMXTRS

PURPOSE - Compute Beam Stresses

CALL SEQUENCE - CALL BMXTRS(II,XL)

ARGUMENTS - II = Element Number

XL = Length

CALLED BY - PRNTEL

EXTERNALS - BMGEØM

COMMON BLOCKS - CØNN, ELEM, FRDR, MAT, N, NØDES, TTBME, AMAXMIN, BASEA, AFK

FILE NAMES - None

NOTES - None

ROUTINE NAME - BØUND2

PURPOSE - Apply Boundary Conditions to the Mass and Stiffness

CALL SEQUENCE - CALL BOUND2

ARGUMENTS - None

CALLED BY - ANALYZ

EXTERNALS - PUTSK, SK

COMMON BLOCKS - BND, EIGN, GM, N, SKGM

FILE NAMES - None

NOTES - The rows and columns in the mass and stiffness matrices which correspond to the boundary degrees of freedom are eliminated.

ROUTINE NAME - CHANGE

PURPOSE - Perform Row and Column Interchanges for Quadrilateral Elements

CALL SEQUENCE - CALL CHANGE(EKM,IX,IY)

ARGUMENTS - EKM = Element Mass or Stiffness Matrix

IX, IY = Row and Column Numbers to be Interchanged

CALLED BY - CØNDNS

EXTERNALS - None

COMMON BLOCKS - None

FILE NAMES - None

NOTES - The rows and columns of a quadrilateral element mass or stiffness matrix must be interchanged so that the degrees of freedom are in ascending order as required by ASEMBL.

ROUTINE NAME - CØNDNS

PURPOSE - Condense the Quad. Mass and Stiffness Matrices

CALL SEQUENCE - CALL CØNDNS(MB,MC,MD,NØ)

ARGUMENTS - MB, MC, MD = Node Numbers

NØ = Control Parameter

CALLED BY - QDRLTL

EXTERNALS - CHANGE

COMMON BLOCKS - EIGN, MASS, STIFF

FILE NAMES - None

NOTES - Since the quadrilateral elements are formed by assembly of four triangles, the center node degrees of freedom must be reduced. The 10 x 10 mass and stiffness matrices are reduced to 8 x 8 by static condensation.

ROUTINE NAME	-	CØØRD
PURPOSE	-	Determine Direction Cosines for Element Transformation
CALL SEQUENCE	-	CALL COORD(L)
ARGUMENTS	-	L = Element Number
CALLED BY	-	GETDK, PRNTEL, STIFFK, STRCØN, UNTFRC
EXTERNALS	-	SQRT
COMMON BLOCKS	-	AREA, CØNN, ELEM, LØCAL, NODES
FILE NAMES	-	None
NOTES	-	None

ROUTINE NAME	-	CRAMER
PURPOSE	-	Cramer's Rule for Inversion of Triangle Coordinate Matrix
CALL SEQUENCE	-	CALL CRAMER(A,TRIANG,MA,MB,MC)
ARGUMENTS	-	A = 3 x 3 Matrix which Contains the Inverse
		TRIANG = Area
		MA, MB, MC = Node Numbers

CALLED BY	-	PLSTIF, STRESS
EXTERNALS	-	None
COMMON BLOCKS	-	LOCAL
FILE NAMES	-	None

NOTES	-	None

ROUTINE NAME	-	CTYPE
PURPOSE	-	Transformations to Cartesian Coordinates
CALL SEQUENCE	-	CALL CTYPE(ITYPE,X1,X2,X3)
ARGUMENTS	-	ITYPE=1 Cylindrical coordinates given =2 Spherical coordinates given X1,X2,X3 = r,θ,z for ITYPE=1 = r,θ,ϕ for ITYPE=2
CALLED BY	-	INXYZ
EXTERNALS	-	None
COMMON BLOCKS	-	None
FILE NAME	-	None
NOTES	-	Other transformations can be supplied by the user. On return, X1,X2,X3 are the x,y,z cartesian coordinates.

ROUTINE NAME	-	CURRENT
PURPOSE	-	Evaluate Current Design and Call Related Routines
CALL SEQUENCE	-	CALL CURRENT
ARGUMENTS	-	None

CALLED BY	-	DØPRØB, LAYPR
EXTERNALS	-	ANALYZ, DAMAGE, DMGFREQ, SCALE, SIVIB2, WEIGHTS
COMMON BLOCKS	-	BASEA, DMG, EIGN, ELEM, FRDR
FILE NAMES	-	TAPE6

NOTES - The current design is evaluated from a stress and deflection standpoint. If required, the natural frequencies and modes will be calculated. The structural weight is also computed.

ROUTINE NAME	-	DAMAGE
PURPOSE	-	Perform Static Reanalysis for Damage Cases
CALL SEQUENCE	-	CALL DAMAGE(L,D)
ARGUMENTS		L = Load Case Counter
		D = Array of Deflections for Each of the Damage Cases

CALLED BY	-	CURRENT, DMØDE
EXTERNALS	-	DMGITER, ERR, GETDK, SECØND
COMMON BLOCKS	-	DMG, DMGERR, EIGVEC, N, TITLE
FILE NAMES	-	TAPE6

NOTES	-	Main iteration loop contains the Aitken's extrapolation procedure. Time and error summary printed at the end of the loop.

ROUTINE NAME - DECØUP

PURPOSE - Decouple the Eigenvector Matrix

CALL SEQUENCE - CALL DECØUP

ARGUMENTS - None

CALLED BY - SIVIB2

EXTERNALS - SQRT

COMMON BLOCKS- EIGN, EIGVEC, N

FILE NAMES - None

NOTES - The eigenvalues are also sorted in descending order of magnitude. The two most recent eigenvectors are decoupled.

ROUTINE NAME	-	DMGFREQ
PURPOSE	-	Frequency Reanalysis for Damage Cases
CALL SEQUENCE	-	CALL DMGFREQ
ARGUMENTS	-	None

CALLED BY	-	CURRENT
EXTERNALS	-	DMGØRTH, ERR, FRQITER, GETDK, SECØND
COMMON BLOCKS	-	DMG, DMGERR, EIGN, EIGVEC, N, NØDES, TITLE
FILE NAMES	-	TAPE6

NOTES - The damaged stiffness and mass matrices are obtained and the vector estimate is orthogonalized with respect to the modified stiffness matrix. Procedure is repeated until the vector error in the norm is small.

ROUTINE NAME - DMGITER

PURPOSE - Damage Reanalysis Iteration

CALL SEQUENCE - CALL DMGITER(X,Y,D,L)

ARGUMENTS - X = Old Iterate

Y = New Iterate

D = Original Displacement Vectors

L = Load Case Counter

CALLED BY - DAMAGE

EXTERNALS - BACKSUB, DX, FØRSUB

COMMON BLOCKS - DK, N

FILE NAMES - None

NOTES - Iteration formula is

$$K\, dx_{new} = dK\, \bar{x}_{old}$$

$$\bar{x}_{new} = x + d\bar{x}_{new} = \text{response in damage condition}$$

where x is the solution of the undamaged problem $Kx = f$. If loads change in damage condition $\bar{f} = f - f_1$, then modify the Y vectors output just after call to DX. Set $Y = Y - f_1$ before call to FØRSUB.

ROUTINE NAME - DMGØRTH

PURPOSE - Orthogonalize Current Eigenvector Estimates

CALL SEQUENCE - CALL DMGØRTH(T)

ARGUMENTS - T = Array of Vectors to be Orthogonalized

CALLED BY - DMGFREQ, FRQITER

EXTERNALS - DX, PRELT, SQRT

COMMON BLOCKS - DK, EIGN, EIGVEC, N

FILE NAMES - None

NOTES - The current eigenvector estimates for the damage cases are orthogonalized with respect to K-dK where $K = LL^T$.

ROUTINE NAME - DMØDE

PURPOSE - Resize in the Displacement Mode

CALL SEQUENCE - CALL DMØDE

ARGUMENTS - None

CALLED BY - DØPRØB

EXTERNALS - BACKSUB, DAMAGE, FØRSUB, LMSIZE, REDUCE, RESTØR, SQRT, UN

COMMON BLOCKS - AMAXMIN, BASEA, DEFLMT, DMG, DUMMY, ELEM, FRDR, LAYERD, LM

FILE NAMES - TAPE6

NOTES - A check is first made to see if there are any active displacements (i.e. ones sufficiently close to their limits). Next a dummy loads vector is formed for each active displacement. The virtual displacements due to these dummy loads are then computed. Static reanalyses for the damage cases are then performed.

ROUTINE NAME	-	DØPRØB
PURPOSE	-	Overall Sequence Control
CALL SEQUENCE	-	CALL DØPRØB
ARGUMENTS	-	None

CALLED BY	-	Main Program
EXTERNALS	-	INITP, CURRENT, EMØDE, SAVE, GETBEST, DMØDE, AVERAGE, PRINT
COMMON BLOCKS	-	LMTEXCD, ØPT, TITLE, WEIGHT

NOTES - Routines are called which read and check the input data. The current design is then analyzed. Resizing is performed in the energy mode and then in the displacement mode. If the weight increases during the next energy mode resizing the best design is returned and the program proceeds with the next step in the optimization. If the weight increases during the displacement mode resizing, this design is either averaged with the current least weight design or resizing is terminated if the weight has more than doubled.

ROUTINE NAME - DX

PURPOSE - Premultiplication by Damaged Mass or Stiffness Matrix

CALL SEQUENCE - CALL DX(Y,D,Z,N)

ARGUMENTS - Y = Vectors Resulting from Premultiplication

D = Damaged Mass or Stiffness Matrix

Z = Vector Premultiplied by D

N = Number of Vectors

CALLED BY - DMGITER, DMGØRTH, FRQITER, GENMSS

EXTERNALS - REDUCE, RESTØR

COMMON BLOCKS - DK, N

FILE NAMES - None

NOTES - None

ROUTINE NAME - ECHØ

PURPOSE - Echo the Input Data

CALL SEQUENCE - CALL ECHØ

ARGUMENTS - None

CALLED BY - INITIAL

EXTERNALS - EØF

COMMON BLOCKS - None

FILE NAMES - TAPE5, TAPE6

NOTES - The input data is printed at the beginning of each run from TAPE5. Fifty lines are printed per page with column and line numbers.

ROUTINE NAME - ELEMIN

PURPOSE - Read Element Data

CALL SEQUENCE - CALL ELEMIN

ARGUMENTS - None

CALLED BY - INPT

EXTERNALS - INP03,PPHDR

COMMON BLOCKS - BASEA, NMAT

FILE NAMES - TAPE5

NOTES - Read element control data and call INP03 which reads connectivity and materials data. Several key variables are defined:

NMAT = NMT = total number of materials

ISØTRN = NMAT-NCØMP = number of isotropic materials

MEMBS = NELEM = number of elements

ROUTINE NAME - ELFØRC

PURPOSE - Get Element Displacements and Transform to Local System

CALL SEQUENCE - CALL ELFØRC(DR,EDR,L,LDS)

ARGUMENTS - DR = Global Displacement Array

EDR = Local Displacement Array

L = Element Number

LDS = Number of Load Cases

CALLED BY - PRNTEL, STRCØN, UNTFRC

EXTERNALS - None

COMMON BLOCKS - CØNN, ELEM, LØCAL, N

FILE NAMES - None

NOTES - None

ROUTINE NAME - ELSTIC

PURPOSE - Generate Elastic Constant Matrix

CALL SEQUENCE - CALL ELSTIC

ARGUMENTS - None

CALLED BY - GETDK, STIFFK, TRQDSTR, UNTFRC

EXTERNALS - None

COMMON BLOCKS - MATAXIS, TEMP

FILE NAMES - None

NOTES - None

ROUTINE NAME	-	ELSTIF
PURPOSE	-	Generate Element Mass and Stiffness for Bars
CALL SEQUENCE	-	CALL ELSTIF(AE)
ARGUMENTS	-	AE = AE for Bars

CALLED BY	-	GETDK, STIFFK
EXTERNALS	-	LMPRØD
COMMON BLOCKS	-	AREA, EIGN, LØCAL, MASS, N, STIFF, TEMP
FILE NAMES	-	None

NOTES	-	The mass terms are divided by 386 for dimensional consistency with the units of the material density that are input.

ROUTINE NAME - EMØDE

PURPOSE - Resize in the Energy Mode

CALL SEQUENCE - CALL EMØDE

ARGUMENTS - None

CALLED BY - DØPRØB

EXTERNALS - LMSIZE, SQRT

COMMON BLOCKS - AMAXMIN, BASEA, ELEM, LAYERD, MAT, STRNENG, TITLE

FILE NAMES - TAPE6

NOTES - Resize according to

$$AE_{new} = AE_{old}\sqrt{U/(RHO*AE_{old}*L)}$$

U = Strain Energy

RHO = Density

L = Length

ROUTINE NAME	-	ERR
PURPOSE	-	Compute an Error Estimate Between S and T
CALL SEQUENCE	-	CALL ERR(S,T,ERRMAX,L,IMAX)
ARGUMENTS	-	S,T = Vectors to be Tested
		ERRMAX = Norm of Error Vector
		L = Number of Vectors to be Tested
		IMAX = Index of the L Vectors which has the Largest Error
CALLED BY	-	DAMAGE, DMGFREQ
EXTERNALS	-	SQRT
COMMON BLOCKS	-	N
FILE NAMES	-	None
NOTES	-	None

ROUTINE NAME	-	ERRØR
PURPOSE	-	Lock Eigenvectors that have Converged within Tolerance
CALL SEQUENCE	-	CALL ERRØR
ARGUMENTS	-	None
CALLED BY	-	SIVIB2
EXTERNALS	-	SQRT
COMMON BLOCKS	-	EIGN, EIGVEC, N
FILE NAMES	-	None
NOTES	-	Eigenvector predictions are locked when the corresponding error is less than TOLVEC (see input instructions) and higher eigenvector have also been locked.

ROUTINE NAME	-	FØRSUB
PURPOSE	-	Forward Substitution Routine
CALL SEQUENCE	-	CALL FØRSUB(V,X,I0,I1)
ARGUMENTS	-	V = Solution Vector
		X = Right Hand Side
		I0 = First Vector in X to Use
		I1 = Last Vector in X to Use

CALLED BY	-	ANALYZ, DMGITER, DMØDE, FRQITER, SIVIB2
EXTERNALS	-	SK
COMMON BLOCKS	-	N, SKGM
FILE NAMES	-	None

NOTES - None

ROUTINE NAME - FRQITER

PURPOSE - Dynamic Reanalysis Iteration

CALL SEQUENCE - CALL FRQITER(IP1)

ARGUMENTS - IP1 = Damage Case Counter

CALLED BY - DMGFREQ

EXTERNALS - BACKSUB, DMGØRTH, DX, FØRSUB, PREMULT

COMMON BLOCKS - DK, EIGN, EIGVEC, N

FILE NAMES - None

NOTES - See Vol. I for a discussion of the dynamic reanalysis iteration.

ROUTINE NAME - GETBC

PURPOSE - Input Boundary Conditions

CALL SEQUENCE - CALL GETBC

ARGUMENTS - None

CALLED BY - INPT

EXTERNALS - None

COMMON BLOCKS - BND, N, TITLE

FILE NAMES - TAPE5, TAPE6

NOTES - See input instructions for a discussion of boundary conditions.

ROUTINE NAME	-	GETBEST
PURPOSE	-	Returns to Previous Best Design
CALL SEQUENCE	-	CALL GETBEST
ARGUMENTS	-	None

CALLED BY	-	DØPRØB
EXTERNALS	-	None
COMMON BLOCKS	-	BASEA, ELEM, LAYERD, SAVE, TITLE
FILE NAMES	-	TAPE6

NOTES	-	In the energy mode, a weight increase results in the lower weight design returned. If there are displacement constraints, GETBEST returns the lower weight design if weight has more than doubled.

ROUTINE NAME	-	GENMSS
PURPOSE	-	Compute Generalized Masses
CALL SEQUENCE	-	CALL GENMSS
ARGUMENTS	-	None

CALLED BY	-	PRNTMDS
EXTERNALS	-	DX, GETDK, PREMULT
COMMON BLOCKS	-	DK, DMG, EIGN, EIGVEC, GENM, N
FILE NAMES	-	None

NOTES	-	The generalized masses are computed in both the damaged and undamaged conditions. The units are converted to pound-force.

ROUTINE NAME	-	GETDK
PURPOSE	-	Set Up the Damaged Stiffness and Mass Matrices
CALL SEQUENCE	-	CALL GETDK(M)
ARGUMENTS	-	M = Damage Case Counter
CALLED BY	-	DAMAGE, DMGFREQ, GENMSS
EXTERNALS	-	ASEMDK, CØØRD, ELSTIC, ELSTIF, LMPMAS, PLMASS, PLSTIF, PREPAR, QDRLTL, TRECØN, TRNSFM
COMMON BLOCKS	-	AREA, DK, DMG, EIGN, ELEM, FRDR, MASS, STIFF
FILE NAMES	-	None
NOTES	-	None

ROUTINE NAME	-	INCØNN
PURPOSE	-	Read Connectivity Data
CALL SEQUENCE	-	CALL INCØNN
ARGUMENTS	-	None

CALLED BY	-	INP03
EXTERNALS	-	NØDCHCK, PPELEM
COMMON BLOCKS	-	AMAXMIN, BASEA, CØNN, ELEM, TITLE
FILE NAMES	-	TAPE5, TAPE6

NOTES - Read connectivity data and generate element connectivity under certain circumstances (see input instructions). The node sequence is checked by calling NØDCHCK and the element connectivity is printed.

ROUTINE NAME	-	INDMGE
PURPOSE	-	Read Damage Input
CALL SEQUENCE	-	CALL INDMGE
ARGUMENTS	-	None

CALLED BY	-	INPT
EXTERNALS	-	None
COMMON BLOCKS	-	BASEA, DMG, DMGERR, TITLE
FILE NAMES	-	TAPE5, TAPE6

NOTES - Mass and stiffness damage factors are read for each element in each damage case.

ROUTINE NAME - INDSPL

PURPOSE - Read Displacement Constraint Data

CALL SEQUENCE - CALL INDSPL

ARGUMENTS - None

CALLED BY - INPT

EXTERNALS - None

COMMON BLOCKS - DEFLMT, N, NØDES, ØPT, TITLE

FILE NAMES - TAPE5, TAPE6

NOTES - Deflection limits for all nodes or selected nodes are input and printed.

ROUTINE NAME - INLAYR

PURPOSE - Read Composite Layer Data

CALL SEQUENCE CALL INLAYR

ARGUMENTS - None

CALLED BY - INPT

EXTERNALS - None

COMMON BLOCKS - AMAXMIN, ANG, BASEA, ELEM, LAYERD, LAYMIN, TITLE

FILE NAMES - TAPE5, TAPE6

NOTES - The fiber directions are read, initial layer proportions are computed, and data is printed.

ROUTINE NAME	-	INLØADS
PURPOSE	-	Read Loads Data
CALL SEQUENCE	-	CALL INLØADS
ARGUMENTS	-	None

CALLED BY	-	INPT
EXTERNALS	-	None
COMMON BLOCKS	-	DMG, DRDR, N. NØDES, TITLE
FILE NAMES	-	TAPE5, TAPE6

NOTES	-	Applied loads are input and printed. Total forces and moments are computed for each load condition.

ROUTINE NAME - INITIAL

PURPOSE - Sets Constants

CALL SEQUENCE - CALL INITIAL

ARGUMENTS - None

CALLED BY - Main Program

EXTERNALS - ECHO, ØPENSK

COMMON BLOCKS - LMTEXCD, NPRØB, QUAD, PØSTPR

FILE NAMES - TAPE5

NOTES - The number of problems is read, a call to ECHØ prints the data listing, and the constants NFIL, MAXSK, NACTIVE, MAA, MBB, MCC are set.

ROUTINE NAME	-	INITP
PURPOSE	-	Problem Initialization
CALL SEQUENCE	-	CALL INITP
ARGUMENTS	-	None

CALLED BY	-	DØPRØB
EXTERNALS	-	INPT, PØP
COMMON BLOCKS	-	NPROB, ØPT, SKGM, TITLE, WEIGHT
FILE NAMES	-	TAPE6

NOTES - This input initialization routine is called at the beginning of each problem. Calls are made to the routine which reads the input and to the routine which maps the stiffness matrix. Variables NPAGE, NECYCL, NDCYCL, and WTLAST are set.

ROUTINE NAME	-	INGNRL
PURPOSE	-	Read General Analysis and Optimization Data
CALL SEQUENCE	-	CALL INGNRL
ARGUMENTS	-	None
CALLED BY	-	INPT
EXTERNALS	-	None
COMMON BLOCKS	-	AMAXMIN, ANG, DEFLMT, DMG, EIGN, LAYERD, LAYMIN, LMASS, N, ØPT, PØSTPR, STRNENG, TITLE
FILE NAMES	-	TAPE5, TAPE6
NOTES	-	The general input is read which controls the program options. This information is also printed with explanatory headings.

ROUTINE NAME	-	INLPMSS
PURPOSE	-	Read Lumped Mass Data
CALL SEQUENCE	-	CALL INLPMSS
ARGUMENTS	-	None
CALLED BY	-	INPT
EXTERNALS	-	None
COMMON BLOCKS	-	LMASS, TITLE
FILE NAMES	-	TAPE5, TAPE6
NOTES	-	None

ROUTINE NAME	-	INP03
PURPOSE	-	Read Element Data
CALL SEQUENCE	-	CALL INP03
ARGUMENTS	-	None

CALLED BY	-	ELEMIN
EXTERNALS	-	PPHDR, INCØNN
COMMON BLOCKS	-	BASEA, MAT, NMAT, TITLE
FILE NAMES	-	TAPE5, TAPE6

NOTES	-	Read materials properties and allowables. Call INCØNN to read the connectivity. Materials properties and allowables are printed.

ROUTINE NAME	-	INPT
PURPOSE	-	Calls all the Input Routines
CALL SEQUENCE	-	CALL INPT
ARGUMENTS	-	None
CALLED BY	-	INIPT
EXTERNALS	-	INGNRL, INXYZ, ELEMIN, INLAYR, GETBC, INLØADS, INDSPL, INDMGE, INLPMSS
COMMON BLOCKS	-	BND, DMG, FRDR, LAYERD, LMASS, N, NØDES, ØPT
FILE NAMES	-	None
NOTES	-	Various input routines are called, depending on the input read in the general input routine INGNRL.

ROUTINE NAME	-	INXYZ
PURPOSE	-	Read Nodal Data
CALL SEQUENCE	-	CALL INXYZ
ARGUMENTS	-	None

CALLED BY	-	INPT
EXTERNALS	-	MESHG, CTYPE, PPHDR, PPNODE
COMMON BLOCKS	-	NØDES, TITLE
FILE NAMES	-	TAPE5, TAPE6

NOTES	-	Nodal data is read, optional mesh generator called, coordinates transformed to cartesian form from cylindrical or spherical if necessary (call to CTYPE), and call routines to write header and nodal data to the optional post-processor file.

ROUTINE NAME - LAYCALC

PURPOSE - Calculate No. of Layers in Each Fiber Direction

CALL SEQUENCE - CALL LAYCALC(L)

ARGUMENTS - L = Element Number

CALLED BY - LAYPR

EXTERNALS - None

COMMON BLOCKS - AREA, BASEA, ELEM, LAYERD, LAYMIN, LAYPRNT, SAVE

FILE NAMES - None

NOTES - None

ROUTINE NAME - LAYPR

PURPOSE - Print # Layers Each Direction, Integerize Design

CALL SEQUENCE - CALL LAYPR

ARGUMENTS - None

CALLED BY - PRINT

EXTERNALS - CURRENT, LAYCALC

COMMON BLOCKS - BASEA, ELEM, LAYERD, LAYMIN, LAYPRNT, OPT, SAVE, TITLE

FILE NAMES - TAPE6

NOTES - None

ROUTINE NAME - LLT

PURPOSE - LLT Decomposition of Stiffness Matrix

CALL SEQUENCE - CALL LLT

ARGUMENTS - None

CALLED BY - ANALYZ

EXTERNALS - PRNTSK, PUTSK, SK, SQRT

COMMON BLOCKS - N, SKGM

FILE NAMES - TAPE6

NOTES - The stiffness matrix is decomposed by the Choleski method into the form $K=LL^T$. The skyline storage space for K is replaced with L^T.

ROUTINE NAME - LMPMAS

PURPOSE - Compute Local Mass Matrix for Triangles and Quads

CALL SEQUENCE - CALL LMPMAS(L)

ARGUMENTS - L = Element Number

CALLED BY - GETDK, STIFFK

EXTERNALS - None

COMMON BLOCKS - AREA, CØNN, ELEM, LØCAL, MASS, TEMP

FILE NAMES - None

NOTES - None

ROUTINE NAME - LMPRØD

PURPOSE - Compute Mass Matrix for Bar

CALL SEQUENCE - CALL LMPRØD(AE)

ARGUMENTS - AE = AE for Bar

CALLED BY - ELSTIF

EXTERNALS - SQRT

COMMON BLOCKS - AREA, LØCAL, MASS, N, TEMP

FILE NAMES - None

NOTES - None

ROUTINE NAME - LMSIZE

PURPOSE - Compute Proportions of Fiber Directions

CALL SEQUENCE - CALL LMSIZE

ARGUMENTS - None

CALLED BY - DMØDE, EMØDE

EXTERNALS - SQRT

COMMON BLOCKS - BASEA, ELEM, LAYENG, LAYERD, LAYMIN

FILE NAMES - None

NOTES - The proportions are also adjusted to satisfy minimum size contraints.

ROUTINE NAME - ØPENSK

PURPOSE - Sets Up Mass Storage for Stiffness Array

CALL SEQUENCE - CALL ØPENSK

ARGUMENTS - NONE

CALLED BY - INITIAL

EXTERNALS - ØPENMS, WRITMS

COMMON BLOCKS - SKBUF

FILE NAMES - None

NOTES - A mass storage file is set up to be used as a pseudo array for the stiffness array SK. This array is divided into MAXREC records of 1000 words per record. Five records are in core at any one time. A list of the record numbers for these five records is contained in the INDXSK array. The SK buffer SKBUF contains the last five SK records. This is done to minimize the MS reads and writes.

ROUTINE NAME - ØRTHØG

PURPOSE - Orthonormalize W

CALL SEQUENCE - CALL ØRTHØG(W,LØCK,L1,NTRIAL)

ARGUMENTS -
W = Vectors to be Orthonormalized
LØCK = Number of Vectors Locked
L1 = Total Number of Vectors Less LØCK
NTRIAL = Number of Trial Vectors

CALLED BY - SIVIB2

EXTERNALS - SQRT

COMMON BLOCKS - N

FILE NAMES - None

NOTES - The "unlocked" trial vectors are orthonormalized by the standard Gram-Schmidt process.

ROUTINE NAME	-	MESHG
PURPOSE	-	Generate Nodal Data
CALL SEQUENCE	-	CALL MESHG(NWORK,MGEN)
ARGUMENTS	-	NWØRK, work space dimension MGEN, generator option (positive number)

CALLED BY	-	INXYZ
EXTERNALS	-	See notes
COMMON BLOCKS	-	NØDES, see notes
FILE NAME	-	See notes

NOTES	-	This is a user written subroutine. In the current version of ADDRESS, no mesh generator is supplied. The routine may contain additional common blocks, externals, and references to files.

ROUTINE NAME - NØDCHCK

PURPOSE - Check Connectivity Data

CALL SEQUENCE - CALL NØDCHCK(IELNØ,ITYPE,NØD,DØNE,ERR)

ARGUMENTS - IELNØ = element number

ITYPE = element type

NØD(4) = element nodes for this element

DØNE = logical error flag (warning)

ERR = logical error flag (fatal)

CALLED BY - INCØNN

EXTERNALS - None

FILE NAMES - TAPE6

NOTES - Connectivity data checked to make sure

MA<MB (bars)

MA<MB<MC (triangle)

MA<min(MB,MC,MD) (quadrilaterals)

Data is corrected for the bars and triangles. Error flag ERR is set to true and the program stops if there is an error in the quadrilateral data.

ROUTINE NAME	-	PLMASS
PURPOSE	-	Determine Mass Matrix for Triangular Elements in the Local Coordinates
CALL SEQUENCE	-	CALL PLMASS(EEMM,AREA)
ARGUMENTS	-	EEMM = Mass Matrix Array
		AREA = Area of Element
CALLED BY	-	GETDK, QDRLTL, STIFFK
EXTERNALS	-	None
COMMON BLOCKS	-	AREA, TEMP
FILE NAMES	-	None
NOTES	-	None

ROUTINE NAME - PLSTIF

PURPOSE - Determine Stiffness Matrix for Triangular Elements in the Local Coordinates

CALL SEQUENCE - CALL PLSTIF(EEKK,AREA,MA,MB,MC,NØNØRM)

ARGUMENTS -
- EEKK = Stiffness Matrix Array
- AREA = Area of Element
- MA,MB,MC = Node Numbers for Elements
- NØNØRM = 0 for membrane elements
  = 1 for shear panel calculations

CALLED BY - GETDK, QDRLTL, STIFFK, UNTFRC

EXTERNALS - CRAMER

COMMON BLOCKS - AREA, MATAXIS

FILE NAMES - None

NOTES - None

ROUTINE NAME - PPDISP

PURPOSE - Write Displacements to Post-processor File 99

CALL SEQUENCE - CALL PPDISP(IL)

ARGUMENTS - IL = Load Case Index

CALLED BY - PRNTDR

EXTERNALS - None

COMMON BLOCKS - FRDR, N, NØDES, PØSTPR

FILE NAMES - NFIL

NOTES - None

ROUTINE NAME	-	PPELEM
PURPOSE	-	Write Element Connectivity to Post-processor File
CALL SEQUENCE	-	CALL PPELEM
ARGUMENTS	-	NONE
CALLED BY	-	INCØNN
EXTERNALS	-	None
FILE NAMES	-	NFIL
NOTES	-	None

```
ROUTINE NAME     -   PPHDR
PURPOSE          -   Write Header to Post-processor File
CALL SEQUENCE    -   CALL PPHDR(ITYPE,I1,I2,I3,I4,R1)
ARGUMENTS        -   ITYPE=1  geometry
                          =2  connectivity
                          =3  end of data
                          =4  nodal displacements
                          =5  element stress
                          =6  element type
                          =7  element data
                          =8  end of problem trailer

                       I1= number of nodes, ITYPE=1
                         = element type, ITYPE=2,6
                         = load case, ITYPE=4
                         = number of element types, ITYPE=5
                         = element number, ITYPE=7

                       I2= max. no. of nodes/element, ITYPE=2
                         = number of nodes, ITYPE=4
                         = load case number, ITYPE=5
                         = dimensionality, ITYPE=6
                         = max. number of nodes/element, ITYPE=7

                       I3= number of nodes, ITYPE=2
                         = interpolation code, ITYPE=6
                         = integration order, ITYPE=7

                       I4= number of element, ITYPE=6

                       R1= load parameter, ITYPE=4,5

CALLED BY        -   ELEMIN, INPO3,INXYZ,PRINT,PRNTDR
EXTERNALS        -   None
COMMON BLOCKS    -   PØSTPR
FILE NAMES       -   NFIL

NOTES            -   None
```

ROUTINE NAME - PPMØDE

PURPOSE - Write Mode Shapes to Post-processor File 8

CALL SEQUENCE - CALL PPMØDE

ARGUMENTS - None

CALLED BY - PRNTMDS

EXTERNALS - None

COMMON BLOCKS - DMG, EIGN, EIGVEC, GENM, N, NØDES, TITLE

FILE NAMES - TAPE8

NOTES - Mode shapes are written to TAPE8. It is assumed that only the Z component of the vector is of interest and that the nodes are numbered in accordance with the standard conventions for a wing structure. Generalized masses, frequencies, and (x,y) locations of the grid points are also output to TAPE8.

ROUTINE NAME	-	PPNØDE
PURPOSE	-	Write Node Coodinates to Post-processor File 99
CALL SEQUENCE	-	CALL PPNØDE
ARGUMENTS	-	None

CALLED BY	-	INXYZ
EXTERNALS	-	None
COMMON BLOCKS	-	NØDE, PØSTPR
FILE NAMES	-	NFIL

NOTES	-	All three components of the coordinates are written to NFIL.

ROUTINE NAME	-	PPTTL
PURPOSE	-	Write Title to Post-processor File
CALL SEQUENCE	-	CALL PPTTL
ARGUMENTS	-	None

CALLED BY	-	INGNRL
EXTERNALS	-	None
COMMON BLOCKS	-	POSTPR, TITLE
FILE NAME	-	NFIL

NOTES	-	The title that is read as input is written to a post-processor file NFIL (set to 99 in INITIAL) for possible later use.

ROUTINE NAME	-	PØP
PURPOSE	-	Map the Stiffness Matrix
CALL SEQUENCE	-	CALL PØP
ARGUMENTS	-	None

CALLED BY	-	INITP
EXTERNALS	-	None
COMMON BLOCKS	-	BASEA, CØNN, ELEM, N, SKGM, TITLE
FILE NAMES	-	TAPE6

NOTES - The gross population (total number of elements in the upper triangle) and the apparent population (total number of elements in the skyline) of the stiffness matrix are computed for the unrestrained structure. The apparent population must be not greater than MAXSK which is defined in INITIAL.

ROUTINE NAME - PRELT

PURPOSE - Compute X = LT*Y

CALL SEQUENCE - CALL PRELT(X,Y,N)

ARGUMENTS - X = Vectors to be Computed

Y = Given Vectors

N = Number of Vectors to be Computed

CALLED BY - DMGØRTH

EXTERNALS - SK

COMMON BLOCKS - N, SKGM

FILE NAMES - None

NOTES - The L matrix is stored in skyline form.

ROUTINE NAME - PREMULT

PURPOSE - Matrix Vector Multiplication: Y = GM*Z

CALL SEQUENCE - CALL PREMULT(Y,Z,N,L)

ARGUMENTS - Y = Vectors to be Computed

Z = Given Vectors

N = Number of Vectors to be Computed

L = Shift Factor for Y

CALLED BY - FRQITER, GENMSS, SIVIB2

EXTERNALS - None

COMMON BLOCKS - GM, N, SKGM

FILE NAMES - None

NOTES - The mass matrix is stored in skyline form.

ROUTINE NAME - PREPAR

PURPOSE - Compute Moduli and Allowable Stresses

CALL SEQUENCE - CALL PREPAR(L,BA,BAE,INDEX,LDS)

ARGUMENTS -
L = Element Number

BA = Reference Thickness Used to Establish Triangle and Quadrilateral Element Thicknesses and Moduli

BAE = Reference Thickness Used to Compute Allowable Stresses

INDEX = 0, Compute Element Thicknesses
= 1, Compute Allowable Stresses

LDS = # of Load Cases

CALLED BY - GETDK, PRNTEL, STIFFK, STRCØN, UNTFRC

EXTERNALS - None

COMMON BLOCKS - ALSTRS, AREA, EIGN, ELEM, ENERGY, ESTRESS, LAYERD, MASS, MAT, MATAXIS, NMAT, STIFF, TEMP

FILE NAMES - None

NOTES -
BA = 1.0 when PREPAR is called by STIFFK, GETDK, STRCØN, and UNTFRC

= BASEA when PREPAR is called by PRNTEL

BAE = 1.0 when PREPAR is called by STIFFK, GETDK, PRNTEL, AND UNTFRC

= BASEA when PREPAR is called by STRCØN

INDEX = 0 when PREPAR is called by STIFFK, GETDK, and UNTFRC

= 1 when PREPAR is called by PRNTEL, STRCØN, and UNTFRC

ROUTINE NAME	-	PRINT
PURPOSE	-	Scale Thicknesses, Call Various Print Routines
CALL SEQUENCE	-	CALL PRINT
ARGUMENTS	-	None

CALLED BY	-	DØPRØB
EXTERNALS	-	LAYPR, PPHDR, PRNTDR, PRNTEL, PRNTMDS
COMMON BLOCKS	-	BASEA, EIGN, ELEM, LAYERD, SAVE
FILE NAMES	-	None

NOTES	-	None

ROUTINE NAME - PRNTDR

PURPOSE - Print Table of Node Information

CALL SEQUENCE - CALL PRNTDR

ARGUMENTS - None

CALLED BY - PRINT

EXTERNALS - PPDISP, PPHDR

COMMON BLOCKS - BND, FRDR, N, NØDES, TITLE

FILE NAMES - TAPE6

NOTES - Coordinates, applied loads, and displacements are printed.

ROUTINE NAME - PRNTEL

PURPOSE - Print Element Information

CALL SEQUENCE - CALL PRNTEL

ARGUMENTS - None

CALLED BY - PRINT

EXTERNALS - CØØRD, ELFØRC, PREPAR, SK, TRQDØUT, TRQDSTR

COMMON BLOCKS - ALSTRS, AREA, BASEA, CØNN, DMG, ELEM, ENERGY, ESTRESS, FRDR, LAYERD, SAVE, STRESS, TEMP, TITLE

FILE NAMES - TAPE6

NOTES - Element thicknesses, connectivity, stresses, and strain energies are printed.

ROUTINE NAME - PRNTSK

PURPOSE - Print the Structural Stiffness or Mass Matrix by Rows

CALL SEQUENCE - CALL PRNTSK(SK)

ARGUMENTS - SK = Pseudo Stiffness Matrix

CALLED BY - LLT

EXTERNALS - SK

COMMON BLOCKS - N, SKGM, TITLE

FILE NAMES - TAPE6

NOTES - This replaces the routine PRINTK in previous versions of this program.

ROUTIN NAME	-	PRNTMDS
PURPOSE	-	Print Out Frequencies and Modeshapes
CALL SEQUENCE	-	CALL PRNTMDS
ARGUMENTS	-	None

CALLED BY	-	PRINT
EXTERNALS	-	ATAN, GENMSS, PPMØDE, RESTØR, SQRT
COMMON BLOCKS	-	DMG, EIGN, EIGVEC, GENM, N, NØDES, PØSTPR, TITLE
FILE NAMES	-	TAPE6

NOTES	-	Before mode shapes are printed they are normalized so that the maximum component is ±1.0.

ROUTINE NAME - PUTSK

PURPOSE - Buffer Bookkeeping

CALL SEQUENCE - CALL PUTSK(IWØRD,VALUE)

ARGUMENTS - See Notes

CALLED BY - ASEMBL, BØUNDS, LLT, STIFFK

EXTERNALS - SK

COMMON BLOCKS - SKBUF

FILE NAMES - None

NOTES - This routine loads the value "VALUE" into the pseudo array SK; i.e. this routine replaces the statement SK(IWØRD) = VALUE.

ROUTINE NAME	-	QDRLTL
PURPOSE	-	Compute Stiffness and Mass Matrices for the Quadrilateral and Shear Panel Elements
CALL SEQUENCE	-	CALL QDRLTL(AREA,L,NØ)
ARGUMENTS	-	AREA = Element Area L = Element Number NØ = Control Parameter (See CØNDNS Arguments)
CALLED BY	-	GETDK, STIFFK, TRQDSTR, UNTFRC
EXTERNALS	-	CØNDNS, PLMASS, PLSTIF, SUM
COMMON BLOCKS	-	AREA, CØNN, EIGN, ELEM, MASS, QUAD, STIFF
FILE NAMES	-	None
NOTES	-	This routine calls the triangular element routines for each of the four elements making up the quadrilateral and makes sure that the results are properly combined into one mass and one stiffness matrix.

ROUTINE NAME	-	QLSTRS
PURPOSE	-	Compute Stresses for 4 Triangles of the Quadrilateral
CALL SEQUENCE	-	CALL QLSTRS(L)
ARGUMENTS	-	L = Element Number
CALLED BY	-	TRQDSTR
EXTERNALS	-	STRESS
COMMON BLOCKS	-	ALSTRS, ELEM, ENERGY, ESTRESS, FRDR, QUAD, STIFF, STRESS
FILE NAMES	-	None
NOTES	-	None

ROUTINE NAME - RANDØM

PURPOSE - Enter Random Vectors into Certain Columns of W

CALL SEQUENCE - CALL RANDØM(W,IFLAG)

ARGUMENTS - W = Initial Array of Eigenvectors
IFLAG = See Notes

CALLED BY - SIVIB2

EXTERNALS - None

COMMON BLOCKS - EIGN, N

FILE NAMES - None

NOTES - If IFLAG=10, random numbers are entered in just the last column of W. When IFLAG=0, random members are generated for all columns of W.

ROUTINE NAME	-	REDUCE
PURPOSE	-	Eliminate Rows from Vector for Boundary DØF
CALL SEQUENCE	-	CALL REDUCE(F,L)
ARGUMENTS	-	F = Vector to be Reduced
		L = Number of Vectors

CALLED BY	-	ANALYZ, DMØDE, DX
EXTERNALS	-	None
COMMON BLOCKS	-	BND, N
FILE NAMES	-	None

NOTES	-	None

ROUTINE NAME - RESTØR

PURPOSE - Restore the Displacement or Force Matrix to Full Size

CALL SEQUENCE - CALL RESTØR(D,L)

ARGUMENTS - D = Vector to be Restored

L = Number of Vectors

CALLED BY - DMIDE, DX, PRNTMDS, SCALE

EXTERNALS - None

COMMON BLOCKS - BND, N

FILE NAMES - None

NOTES - None

ROUTINE NAME	-	SAVE
PURPOSE	-	Save Current Least-weight Design
CALL SEQUENCE	-	CALL SAVE
ARGUMENTS	-	None

CALLED BY	-	DØPRØB
EXTERNALS	-	None
COMMON BLOCKS	-	BASEA, ELEM, LAYERD, SAVE, WEIGHT
FILE NAMES	-	None

NOTES	-	The design is saved if the weight decreased from the previous cycle.

ROUTINE NAME	-	SCALE
PURPOSE	-	Adjust Scaling Factor to Satisfy Constraints
CALL SEQUENCE	-	CALL SCALE
ARGUMENTS	-	None
CALLED BY	-	CURRENT
EXTERNALS	-	None
COMMON BLOCKS	-	AMAXMIN, BASEA, DEFLMT, ELEM, FRDR, LMTEXCD,N, ØPT, STRNE
FILE NAMES	-	None
NOTES	-	Maximum-size constraints are also impared by this subroutine.

ROUTINE NAME - SIVIB2

PURPOSE - Compute Eigenvalues and Eigenvectors

CALL SEQUENCE - CALL SIVIB2

ARGUMENTS - None

CALLED BY - CURRENT

EXTERNALS - BACKSUB, DECØUP, ERRØR, FØRSUB, ØRTHØG, PREMULT, RANDØM, SECØND

COMMON BLOCKS - EIGN, EIGVEC, N

FILE NAMES - TAPE6

NOTES - This is the main calling routine for the eigensolution. See Vol. I for a discussion of the simultaneous iteration techniques used in this routine.

ROUTINE NAME	-	SK
PURPOSE	-	Bookkeeping for Stiffness Matrix, SK
CALL SEQUENCE	-	FUNCTIØN SK(IWØRD)
ARGUMENTS	-	IWØRD = See Notes
CALLED BY	-	ASEMBL, BACKSUB, BØUNDS, FØRSUB, LLT, PRELT, PRNTEL, STRCØN, PUTSK, PRNTSK
EXTERNALS	-	READMS, WRITMS
COMMON BLOCKS	-	SKBUF
FILE NAMES	-	TAPE6
NOTES	-	This function returns the value of SK(IWØRD) from the pseudo array SK. Since SK has been replaced by a MS file, this routine performs all the bookkeeping required to access the correct word on the correct record.

ROUTINE NAME	-	STIFFK
PURPOSE	-	Set Up Total Stiffness and Mas Matrices for Structure
CALL SEQUENCE	-	CALL STIFFK
ARGUMENTS	-	None
CALLED BY	-	ANALYZ
EXTERNALS	-	ASEMBL, CØØRD, ELSTIC, ELSTIF, LMPMAS, PLMASS, PLSTIF, PREPAR, PUTSK, QDRLTL, TRECØN, TRNSFM
COMMON BLOCKS	-	AREA, BASEA, EIGN, ELEM, FRDR, GM, LMASS, MASS, N, SKGM, STIFF, STRNENG
FILE NAMES	-	None
NOTES	-	This is the main calling routine which sets up the global mass and stiffness matrices.

ROUTINE NAME	-	STRCØN
PURPOSE	-	Determine Scale Factor Needed to Satisfy Constraints
CALL SEQUNCE	-	CALL STRCØN
ARGUMENTS	-	None
CALLED BY	-	SCALE
EXTERNALS	-	CØØRD, ELFØRC, PREPAR, SK, TRQDSTR
COMMON BLOCKS	-	ALSTRS, AREA, BASEA, DMG, ELEM, ENERGY, ESTRESS, FRDR, STRESS, STRNENG, TEMP
FILE NAMES	-	TAPE6
NOTES	-	None

ROUTINE NAME	-	STRESS
PURPOSE	-	Compute Strains and Stresses for Triangular Element
CALL SEQUENCE	-	CALL STRESS(UV,MA,MB,MC,ENG,NND)
ARGUMENTS	-	UV = Local Strains
		MA,MB,MC = Node Numbers
		ENG = Element Strain Energy
		NND = 5 If Triangular Element is Part of a Shear Panel
CALLED BY	-	QLSTRS, TRQDSTR
EXTERNALS	-	CRAMER, SQRT
COMMON BLOCKS	-	ALSTRS, AREA, ESTRESS, FRDR, MATAXIS, STRESS
FILE NAMES	-	None
NOTES	-	None

ROUTINE NAME - SUM

PURPOSE - Assemble a Single Matrix from 4 Triangular Matrices

CALL SEQUENCE - CALL SUM(EKM,EEKM,MA,MB,MC)

ARGUMENTS - EKM = Mass or Stiffness Matrix to be Computed

EEKM = Mass or Stiffness Matrices for the Individual Triangular Elements

MA,MB,MC = Node Numbers for the Triangular Element

CALLED BY - QDRLTL

EXTERNALS - None

COMMON BLOCKS - None

FILE NAMES - None

NOTES - None

ROUTINE NAME	-	TRECØN
PURPOSE	-	Coordinate Transformations: Orthotropic Materials
CALL SEQUENCE	-	CALL TRECØN(L,IND)
ARGUMENTS	-	L = Element Number IND = 1 for 0° calculations 2 for 90° calculations 3 for +45° calculations 4 for -45° calculations
CALLED BY	-	GETDK, STIFFK, TRQDSTR, UNTFRC
EXTERNALS	-	CØS, SIN, SQRT
COMMONG BLOCKS	-	ANG, LØCAL, MATAXIS
FILE NAMES	-	None
Notes	-	None

ROUTINE NAME - TRNSFM

PURPOSE - Transform a Matrix From Local to Global Coordinates

CALL SEQUENCE - CALL TRNSFM(EKM,CCC,NNØDES)

ARGUMENTS - EKM = Element Matrix in Local Coordinate

CCC = Global Matrix

NNØDES = 3 For Triangular Element
= 4 For Quad Element

CALLED BY - GETDK, STIFFK

EXTERNALS - None

COMMON BLOCKS - LØCAL, N

FILE NAMES - None

NOTES - None

ROUTINE NAME	-	TRQDØUT
PURPOSE	-	Output--Triangles and Quadrilaterals
CALL SEQUENCE	-	CALL TRQDOUT(L)
ARGUMENTS	-	L = Element Number
CALLED BY	-	PRNTEL
EXTERNALS	-	None
COMMON BLOCKS	-	ALSTRS, AREA, CØNN, ELEM, ENERGY, ESTRESS, FRDR, LAYERD, S
FILE NAMES	-	None
NOTES	-	Print element thickness, stresses, and energy.

ROUTINE NAME - TRQDSTR

PURPOSE - Compute Triangle and Quadrilateral Stress, Energy

CALL SEQUENCE - CALL TRQDSTR(L,BA)

ARGUMENTS - L = Element Number

BA = 1.0 for stresses in element of unit thickness

= BASEA for stresses in current design

CALLED BY - PRNTEL, STRCØN

EXTERNALS - ELSTIC, QDRLTL, QLSTRS, STRESS, TRECØN

COMMON BLOCKS - AREA, DMG, ELEM, ENERGY, ESTRESS, FRDR, LAYENG, LAYMIN, ST

FILE NAMES - None

NOTES - Stresses and engergy for both isotropic and composite elements are computed.

ROUTINE NAME - UNITEG

PURPOSE - Determine Energy Due to Dummy Unit Loads

CALL SEQUENCE - CALL UNITEG(L,II,INDX)

ARGUMENTS - L = Element Number

II = 1, 0° composite larger calculation
2, 90° composite larger calculation
3, +45° composite larger calculation
4, -45° composite larger calculation

INDX = 0 if this is not a composite element

CALLED BY - UNTFRC

EXTERNALS - None

COMMON BLOCKS - AREA, DMG, DUMMY, ELEM, FRDR, LAYENG, LMTEXCD, STIFF, STRNENG

FILE NAMES - None

NOTES - None

ROUTINE NAME	-	UNTFRC
PURPOSE	-	Determine Stress Due to Dummy Unit Loads
CALL SEQUENCE	-	CALL UNTFRC
ARGUMENTS	-	None

CALLED BY	-	DMØDE
EXTERNALS	-	CØØRD, ELFØRC, ELSTIC, PLSTIF, PREPAR, QDRLTL, TRECØN, UNITEG
COMMON BLOCKS	-	ANG, AREA, BASEA, DUMMY, ELEM, FRDR, LMTEXCD, STIFF
FILE NAMES	-	None

NOTES	-	None

ROUTINE NAME - WEIGHTS

PURPOSE - Determine Weights for Structure, Scale Displacements

CALL SEQUENCE - CALL WEIGHTS

ARGUMENTS - None

CALLED BY - CURRENT

EXTERNALS - None

COMMON BLOCKS - BASEA, EMG, ELEM, FRDR, LMASS, MAT, N, NPRØB, ØPT, WEIGHT

FILE NAMES - TAPE6

NOTES - Total structural weight as well as weights of all the elements by type are computed and printed.

APPENDIX D

UPDATES FOR OUT OF CORE SOLUTIONS

```
*ID DELETE1
*D RFTDAMOPT.52
      COMMON /GM/ GM(1)
*D RFTDAMOPT.236
      COMMON /GM/ GM(1)
*D RFTDAMOPT.2946
      COMMON /GM/ GM(1)
*D RFTDAMOPT.3707
      COMMON /GM/ GM(1)
*D RFTDAMOPT.555
      COMMON /EIGVEC/ U(350,6),V(350,6),W(1,1,1),XL(10,3),Y(1,1)
*D RFTDAMOPT.646
      COMMON /EIGVEC/ U(350,6),V(350,6),W(1,1,1),XL(10,3),Y(1,1)
*D RFTDAMOPT.718
      COMMON /EIGVEC/ U(350,6),V(350,6),W(1,1,1),XL(10,3),Y(1,1)
*D RFTDAMOPT.799
      COMMON /EIGVEC/ U(350,6),V(350,6),W(1,1,1),XL(10,3),Y(1,1)
*D RFTDAMOPT.1247
      COMMON /EIGVEC/ U(350,6),V(350,6),W(1,1,1),XL(10,3),Y(1,1)
*D RFTDAMOPT.1305
      COMMON /EIGVEC/ U(350,6),V(350,6),W(1,1,1),XL(10,3),Y(1,1)
*D RFTDAMOPT.1340
      COMMON /EIGVEC/ U(350,6),V(350,6),W(1,1,1),XL(10,3),Y(1,1)
*D RFTDAMOPT.1887
 5140 FORMAT(3F10.3)
*D RFTDAMOPT.2871
      COMMON /EIGVEC/ U(350,6),V(350,6),W(1,1,1),XL(10,3),Y(1,1)
*D RFTDAMOPT.3262
      COMMON /EIGVEC/ U(350,6),V(350,6),W(1,1,1),XL(10,3),Y(1,1)
*D RFTDAMOPT.3633
      COMMON /EIGVEC/ U(350,6),V(350,6),W(1,1,1),XL(10,3),Y(1,1)
*ID BIGSK
*D RFTDAMOPT.3
     &         TAPE99=DPOST,TAPE4)
*I RFTDAMOPT.1836
      CALL OPENSK
*D RFTDAMOPT.55
*D RFTDAMOPT.212
*D RFTDAMOPT.238
*D RFTDAMOPT.1281
*D RFTDAMOPT.2338
*D RFTDAMOPT.2927
*D RFTDAMOPT.3711
*D RFTDAMOPT.106
20        CALL PUTSK(JX,SK(JX)+C(I,J))
*D RFTDAMOPT.272
                CALL PUTSK(KX,SK(KXX))
*D RFTDAMOPT.2343
      CALL PUTSK(1,SQRT(SK(1)))
*D RFTDAMOPT.2357
   4        CALL PUTSK(J+IQ,EL/SK(IDIAG(J)))
*I RFTDAMOPT.2339
      EXTERNAL SK
```

```
*D RFTDAMOPT.2369
      CALL PRNTSK(SK)
*D RFTDAMOPT.3720
 10       CALL PUTSK(I,0.)
*D RFTDAMOPT.2360
           CALL PUTSK(IDIAG(I),SQRT(EL))
*I RFTDAMOPT.4433
      SUBROUTINE OPENSK
C
C THIS ROUTINE SETS UP A MASS STORAGE FILE TO BE USED AS A PSEUDO ARRAY
C FOR THE STIFFNESS ARRAY SK. SK WILL BE DIVIDED INTO MAXREC RECORDS
C OF 1000 WORDS PER RECORD. 5 RECORDS WILL BE IN CORE AT ANY ONE TIME.
C A LIST OF THE RECORD NUMBERS FOR THESE 5 RECORDS IS CONTAINED IN
C THE ARRAY INDXSK. THE MAPPING OF SK ONTO THE MASS STORAGE FILE IS
C AS FOLLOWS
C
C
C      SK(1) THROUGH  SK(1000)      RESIDES ON MS RECORD 1
C      SK(1001)  -    SK(2000)     RESIDES ON MS RECORD 2
C
C       AND SO ON
C
C
C      THE SK BUFFER SKBUF CONTAINS THE LAST 5 SK RECORDS THAT HAVE BEEN
C      ACCESSED. THIS IS DONE SO THEAT,HOPFULLY, THE NUMBER OF MS READS
C      AND WRITES WILL BE MINIMIZED.
C
C                                               J. JENSEN  11/1/81
C
      COMMON /SKBUF/ SKBUF(1000,5),INDXSK(5),MAXREC,IDIRSK(101),NXTREC
C
C
      MAXREC = 100
      CALL OPENMS(4,IDIRSK,MAXREC+1,0)
C
C  INITIALIZE THE MASS STORAGE FILE TO DEBUG VALUES
C
      DO 10 I = 1,MAXREC
 10      CALL WRITMS(4,SKBUF(1,1),1000,I,-1)
C
C SET THE BUFFER INDEX TO ALL ZEROES  ( NO RECORDS IN THE SK BUFFER)
C
      DO 20 I = 1,5
 20      INDXSK(I) = 0
      NXTREC = 1
C
C NXTREC POINTS TO THE OLDEST RECORD IN THE BUFFER. THIS RECORD WILL
C BE REPLACED WHEN IT IS NECESSARY TO LOAD A NEW RECORD INTO THE BUFFER.
C
      RETURN
      END
      FUNCTION SK(IWORD)
C
```

```
C THIS FUNCTION RETURNS THE VALUE OF SK(IWORD) FROM THE PSEUDO ARRAY SK.
C SINCE SK HAS BEEN REPLACED BY A MS FILE, THIS ROUTINE PERFORMS ALL
C THE BOOKKEEPING REQUIRED TO ACCESS THE CORRECT WORD ON THE CORRECT
C RECORD
C
      COMMON /SKBUF/ SKBUF(1000,5),INDXSK(5),MAXREC,IDIRSK(101),NXTREC
C
C COMPUTE THE RECORD NUMBER ON WHICH SK(IWORD) RESIDES
C
 5    IREC = (IWORD -1) / 1000 + 1
C
C CHECK TO SEE OF THE RECORD IS ALREADY IN THE BUFFER
C
      DO 10 I = 1,5
          IF(INDXSK(I).NE.IREC) GO TO 10
C
C RECORD IS IN CORE, RETURN THE APPROPRIATE WORD
C
          IW = IWORD - (IREC - 1) * 1000
          SK = SKBUF(IW,I)
          RETURN
 10   CONTINUE
C
C RECORD IS NOT IN THE BUFFER SO RETRIEVE IT
C
      IF(IREC.LE.MAXREC) GO TO 15
           WRITE(6,11) IWORD,IREC,MAXREC
 11        FORMAT(" IN SK IWORD,IREC,MAXREC=",3I14)
           STOP "IREC TO BIG IN SK, ERROR"
15    CONTINUE
C
C FIRST WRITE THE RECORD CURRENTLY IN THE BUFFER BACK TO DISK
C
      IF(INDXSK(NXTREC).NE.0) CALL WRITMS(4,SKBUF(1,NXTREC),1000,
     X                                INDXSK(NXTREC),-1)
C
C AND READ IN THE REQUIRED RECORD
C
      CALL READMS(4,SKBUF(1,NXTREC),1000,IREC)
C
C AND PUT THE RECORD NUMBER IN THE BUFFER DIRECTORY
C
      INDXSK(NXTREC) = IREC
      NXTREC = NXTREC + 1
      IF(NXTREC.GT.5) NXTREC = 1
C
C NOW THAT THE RECORD IS IN THE BUFFER, GO BACK AND RETRIEVE THE
C REQUESTED WORK
C
      GO TO 5
      END
      SUBROUTINE PUTSK(IWORD,VALUE)
C
```

```
C THIS SUBROUTINE LOADS THE VALUE "VALUE" INTO THE PSEUDO ARRAY SK.
C THAT IS THIS ROUTINE REPLACES THE STATEMENT
C           SK(IWORD) = VALUE
C
C
      COMMON /SKBUF/ SKBUF(1000,5),INDXSK(5),MAXREC,IDIRSK(101),NXTREC
C
C FIRST MAKE SURE SK(IWORD) IS IN THE BUFFER BY READING SK(IWORD)
C
      DUMMY = SK(IWORD)
C
C FIND WHICH BUFFER SK(IWORD) IS IN
C
      IREC = (IWORD - 1) / 1000 + 1
      DO 10 I = 1,5
         IF(INDXSK(I).NE.IREC) GO TO 10
         IW = IWORD - (IREC - 1) * 1000
         SKBUF(IW,I) = VALUE
         RETURN
 10      CONTINUE
C
C IF NOT FOUND, AN ERROR HAS OCCURED SOMEWHERE
C
      STOP "ERROR IN PUTSK, RECORD NO IN BUFFER"
      END
      SUBROUTINE PRNTSK(SK)
      COMMON /N/ MM,NM,NN
      COMMON /SKGM/ ICOL(270),IDIAG(270),NONZRO
      COMMON /TITLE/ NPAGE,TITLE(8)
      EXTERNAL SK
C
C     PRINT THE STRUCTURAL STIFFNESS OR MASS MATRIX
C     BY ROWS
C
      WRITE(6,3000) TITLE,NPAGE
      NPAGE=NPAGE+1
      WRITE(6,1000) 1
      WRITE(6,2000) SK(1)
      DO 80 I=2,NM
         KX=IDIAG(I-1)+1
         KY=IDIAG(I)
         WRITE(6,1000)  I
         WRITE(6,2000)  (SK(K),K=KX,KY)
   80 CONTINUE
 1000 FORMAT(2X,I8)
 2000 FORMAT(10X,10E12.4)
 3000 FORMAT(1H1/30X,8A10,10X,5HPAGE ,I3//30X,
     &    *TOTAL STIFFNESS OR MASS MATRIX*)
      RETURN
      END
*D RFTDAMOPT.839
      COMMON /MAT/ ALSTRS(2265),ELCNST(906),POISON(453),RHO1(453),
     X                YOUNGM(453)
```

```
*D RFTDAMOPT.1190
      COMMON /MAT/ ALSTRS(2265),ELCNST(906),POISON(453),RHO1(453),
     X             YOUNGM(453)
*D RFTDAMOPT.2115
      COMMON /MAT/ ALSTRS(2265),ELCNST(906),POISON(453),RHO1(453),
     X             YOUNGM(453)
*D RFTDAMOPT.2981
      COMMON /MAT/ ALSTRS(2265),ELCNST(906),POISON(453),RHO1(453),
     X             YOUNGM(453)
*D RFTDAMOPT.4373
      COMMON /MAT/ ALSTRS(2265),ELCNST(906),POISON(453),RHO1(453),
     X             YOUNGM(453)
*D RFTDAMOPT.1918
      GO TO (90,40,50,60,70,80,85), LAM(I)+1
*INSERT RFTDAMOPT.1936
      GO TO 90
 85   AEX(I)=.50
      AEY(I)=.10
*D RFTDAMOPT.1835
      MAXSK = 100000
*I RFTDAMOPT.2000
                 IF(KY.LE.0) GO TO 370
*D RFTDAMOPT.214
      REAL U(270,I1),W(270,I1)
*D RFTDAMOPT.1283
      REAL V(270,I1),X(270,I1)
*D RFTDAMOPT.3840,3842
                         SX(K)=E1*(EDR(2,K)-EDR(1,K))/AL
                         IF(SX(K).GE.O.) ESRTIO(K)=SK(K)/ALS(1)
                         IF(SX(K).LT.O.) ESRTIO(K)=ABS(SX(K))/ALS(2)
*D RFTDAMOPT.3225,3227
                         SX(K)=E1*(EDR(2,K)-EDR(1,K))/AL
                         IF(SX(K).GE.O.) ESRTIO(K)=SK(K)/ALS(1)
                         IF(SX(K).LT.O.) ESRTIO(K)=ABS(SX(K))/ALS(2)
```

APPENDIX E
TRUSS PROGRAM LISTING

```
      PROGRAM TRUSS
      COMMON /ORDER/ N,NOPTION
      COMMON /YMLV/ E,F(8),OLDF(8),X(10),XB(10)
      CHARACTER*1 ANSWER
C
C -------------------------------------------------------------------
C                                                                   -
C   THIS PROGRAM DOES A DEFLECTION AND STRESS ANALYSIS ON           -
C   A "SIMPLE" TRUSS.                                               -
C                                                                   -
C -------------------------------------------------------------------
C                                                                   -
C  PRINCIPAL VARIABLES:                                             -
C  --------- ---------                                              -
C                                                                   -
C      E          ---->  YOUNG'S MODULUS                            -
C      F(8)       ---->  APPLIED LOAD VECTOR                        -
C      X(10)      ---->  UNMODIFIED MEMBER SIZES                    -
C      XB(10)     ---->  MODIFIED MEMBER SIZES                      -
C      EK(8,8,10) ->  ELEMENT MATRICES                              -
C      SK(8,8) ---->  SYSTEM STIFFNESS MATRIX                       -
C      S(10,8) ---->  STRESS MATRIX                                 -
C      R(8)       ---->  SOLUTION FOR (UN)MODIFIED STRUCTURE        -
C      ST(8)      ---->  ELEMENT STRESSES FOR STRUCTURE             -
C                                                                   -
C -------------------------------------------------------------------
C  DATE: MARCH 18, 1982      KEITH MILLER, PHONE:229-4235           -
C  ----                                        -----                -
C                                                                   -
C -------------------------------------------------------------------
C
C  OPEN INPUT & OUTPUT FILES FOR INTERACTIVE PURPOSES.
C
      OPEN(5,FILE='INPUT')
      OPEN(6,FILE='OUTPUT')
C
C KEEP TRACK OF THE CPU SECONDS USED.
C
      TZERO = SECOND()
      WRITE(6,600)
  600 FORMAT(1HO,' NOPTION',/,1H ,1X,7(1H-),/,1HO,3X,
     &       ' 1        DEFLECTION AND STRESS ANALYSIS ONLY.',/,1HO,
     &  3X,' 2        DEFLECTION AND STRESS ANALYSIS OF',/,1H ,
     &  3X,'          MODIFIED STRUCTURE USING "TAYLOR SERIES".',/,1HO,
     &  3X,' 3        DEFLECTION AND STRESS ANALYSIS OF',/,1H ,
     &  3X,'          MODIFIED STRUCTURE USING SIMPLE ITERATION.',//,1HO,
     &       ' NOPTION.............:')
      READ(5,*) NOPTION
C
C READ IN THE ORDER OF THE STIFFNESS MATRIX.
C
      WRITE(6,605)
  605 FORMAT(1HO,'ORDER OF STIFFNESS MATRIX.....:')
```

```
      READ(5,*) N
      WRITE(6,610)
  610 FORMAT(1H0,'YOUNGS MODULUS...............:')
      READ(5,*) E
C
      WRITE(6,620)
  620 FORMAT(1H0,'INPUT LOAD VECTOR............:')
      READ(5,*) (F(I),I=1,N)
C
C COPY ALL LOADS INTO VECTOR "OLDF".
C THE VECTOR "OLDF" IS NEEDED ONLY WHEN THE USER
C SELECTS THE 3'RD OPTION.
C
      DO 5 I=1,N
    5 OLDF(I) = F(I)
C
      WRITE(6,630)
  630 FORMAT(1H0,'INPUT MEMBER SIZES OF',/,1H ,
     &           'UNMODIFIED STRUCTURE..........:')
      READ(5,*) (X(I),I=1,10)
C
C IFLAG DETERMINES IF THE VECTOR "X" OR "XB" SHOULD
C BE USED FOR THE MEMBER SIZES.
C
C IFLAG = 1 ---> "X" CONTAINS THE CURRENT MEMBER SIZES.
C IFLAG = 2 ---> "XB" CONTAINS THE CURRENT MEMBER SIZES.
C
      IFLAG = 1
C
      CALL GENEK
C
C
   10 CALL GENSK(IFLAG)
C
C GENERATE THE STRESS MATRIX.
C
      CALL GENS
C
C  CHOLESKI DECOMPOSITION OF MATRIX "SK".
C
      CALL LLT
C
C  FORWARD SUBSTITUTION..
C
      CALL FSUB
C
C  BACKWARD SUBSTITUTION.
C
      CALL BSUB
C
C PRINT OUT SOLUTION VECTOR
C
      CALL PRTANS
```

```
C
C  COMPUTE ELEMENT STRESSES.
C
      CALL ESTRESS(IFLAG)
C
C COMPUTE CPU TIME THAT HAS ELAPSED.
C
      ET = SECOND()
      WRITE(6,640) ET-TZERO
  640 FORMAT(1H0,'TIME ELAPSED FROM PROGRAM START = ',F5.2,' CPU SECS.')
C
C DEPENDING UPON THE VALUE OF "NOPTION" PERFORM A DIFFERENT TASK.
C
      GOTO (20,30,40), NOPTION
C
C NOPTION = 1 (DEFLECTION & STRESS ANALYSIS ONLY)
C
   20 CONTINUE

C
C ASK THE USER IF SHE/HE WISHES TO CHANGE THE STRUCTURE SOMEWHAT.
C
      WRITE(6,650)
  650 FORMAT(1H0,'WOULD YOU LIKE TO CHANGE THE MEMBER SIZES? (Y/N) ')
      READ(5,500) ANSWER
  500 FORMAT(A1)
      IF(ANSWER.NE.'Y'.AND.ANSWER.NE.'N') THEN
           WRITE(6,655)
  655      FORMAT(1H0,'SORRY..PLEASE REPEAT YOUR REPLY..."Y"=YES,"N"=NO')
           GOTO 20
      END IF
      IF (ANSWER .EQ. 'Y') THEN
           CALL CHGSIZE(IFLAG)
           IFLAG = 2
           GOTO 10
      ELSE
           STOP 'NOPTION = 1'
      END IF
C
C NOPTION = 2 (DEFLECTION & STRESS ANALYSIS OF MODIFIED STRUCTURE
C                USING 'TAYLOR SERIES'.)
C
   30 CALL CHGSIZE(IFLAG)
      CALL OPT2
      ET = SECOND()
      WRITE(6,640) ET-TZERO
   35 WRITE(6,650)
      READ(5,500) ANSWER
      IF(ANSWER.NE.'Y' .AND. ANSWER.NE.'N') THEN
           WRITE(6,655)
           GOTO 35
      END IF
      IF(ANSWER.EQ.'N') THEN
```

```
            STOP 'NOPTION = 2'
        ELSE
            IFLAG = 2
            GOTO 30
        END IF
C
C NOPTION = 3 (DEFLECTION & STRESS ANALYSIS OF MODIFIED STRUCTURE USING
C               SIMPLE ITERATION.)
C
   40 CONTINUE
C
C OBTAIN INFO CONCERNING THIS OPTION.
C
        CALL OPT3(NITER)
C
C CHANGE CERTAIN MEMBER SIZES.
C
        CALL CHGSIZE(IFLAG)
C
        CALL ITER(NITER)
        ET = SECOND()
        WRITE(6,640) ET-TZERO
   45 WRITE(6,650)
        READ(5,500) ANSWER
        IF(ANSWER.NE.'Y'.AND.ANSWER.NE.'N') THEN
            WRITE(6,655)
            GOTO 45
        END IF
        IF(ANSWER.EQ.'N') THEN
            STOP 'NOPTION = 3'
        ELSE
            IFLAG = 2
            GOTO 40
        END IF
        END
C ********************************************************************
        SUBROUTINE GENEK
        COMMON /ELEMAT/ EK(8,8,10)
        COMMON /ORDER/ N,NOPTION
C
C  GENERATE THE ELEMENT MATRICES.
C
C INITIALIZE ALL ELEMENTS TO ZERO.
C
        DO 5 I=1,N
        DO 5 J=1,N
        DO 5 K=1,10
    5 EK(I,J,K) = 0.0
C
C NOW GENERATE EACH NON-ZERO ELEMENT FOR EACH MATRIX.
C
C MATRIX K1.
C
```

```
      EK(1,1,1) = 1.0
C
C MATRIX K2.
C
      EK(1,1,2) = 1.0
      EK(1,3,2) = -1.0
      EK(3,1,2) = -1.0
      EK(3,3,2) = 1.0
C
C MATRIX K3.
C
      EK(7,7,3) = 1.0
C
C MATRIX K4.
C
      EK(5,5,4) = 1.0
      EK(5,7,4) = -1.0
      EK(7,5,4) = -1.0
      EK(7,7,4) = 1.0
C
C MATRIX K5.
C
      EK(2,2,5) = 1.0
      EK(2,8,5) = -1.0
      EK(8,2,5) = -1.0
      EK(8,8,5) = 1.0
C
C MATRIX K6.
C
      EK(4,4,6) = 1.0
      EK(4,6,6) = -1.0
      EK(6,4,6) = -1.0
      EK(6,6,6) = 1.0
C
C MATRIX K7.
C
C NOTE: T = 1./(2.*SQRT(2.0))
C
      T = 0.35355339
C
      EK(7,7,7) = T
      EK(7,8,7) = T
      EK(8,7,7) = T
      EK(8,8,7) = T
C
C MATRIX K8.
C
      EK(1,1,8) = T
      EK(1,2,8) = -T
      EK(2,1,8) = -T
      EK(2,2,8) = T
C
C MATRIX K9.
```

```
C
      EK(1,1,9) = T
      EK(1,2,9) = T
      EK(2,1,9) = T
      EK(2,2,9) = T
      EK(1,5,9) = -T
      EK(1,6,9) = -T
      EK(2,5,9) = -T
      EK(2,6,9) = -T
      EK(5,5,9) = T
      EK(5,6,9) = T
      EK(6,5,9) = T
      EK(6,6,9) = T
      EK(5,1,9) = -T
      EK(5,2,9) = -T
      EK(6,1,9) = -T
      EK(6,2,9) = -T
C
C MATRIX K10.
C
      EK(3,3,10) = T
      EK(3,4,10) = -T
      EK(4,3,10) = -T
      EK(4,4,10) = T
      EK(3,7,10) = -T
      EK(3,8,10) = T
      EK(4,7,10) = T
      EK(4,8,10) = -T
      EK(7,7,10) = T
      EK(7,8,10) = -T
      EK(8,7,10) = -T
      EK(8,8,10) = T
      EK(7,3,10) = -T
      EK(7,4,10) = T
      EK(8,3,10) = T
      EK(8,4,10) = -T
C
      RETURN
      END
C ***********************************************************************
      SUBROUTINE GENSK(IFLAG)
      COMMON /ELEMAT/ EK(8,8,10)
      COMMON /ORDER/ N,NOPTION
      COMMON /STIFFM/ SK(8,8),R(8)
      COMMON /YMLV/ E,F(8),OLDF(8),X(10),XB(10)
      DIMENSION XSIZE(10)
C
C  FORM STIFFNESS MATRIX:
C
C  INITIALIZE STIFFNESS MATRIX TO ZERO.
C
      DO 10 I=1,N
      DO 10 J=1,N
```

```
   10 SK(I,J) = 0.0
C
C "XSIZE" SHOULD CONTAIN THE MODIFIED MEMBER SIZES,
C IF IFLAG = 1 THEN THE PROGRAM IS PERFORMING THE INITIAL ANALYSIS
C ELSE IFLAG = 2 WHICH INDICATES THAT THE USER HAS MODIFIED THE
C MEMBER SIZES.
C
      IF (IFLAG .EQ. 1) THEN
          DO 15 I=1,10
   15     XSIZE(I) = X(I)
      ELSE
          DO 18 I=1,10
   18     XSIZE(I) = XB(I)
      END IF
C
C  COMPUTE STIFFNESS MATRIX.
C
      DO 20 I=1,10
      DO 20 J=1,N
      DO 20 K=1,N
   20 SK(J,K) = SK(J,K) + XSIZE(I)*EK(J,K,I)
C
      CONST = E/360.
C
      DO 30 I=1,N
      DO 30 J=1,N
   30 SK(I,J) = CONST*SK(I,J)
C
      RETURN
      END
C ***************************************************************
      SUBROUTINE GENS
      COMMON /ORDER/ N,NOPTION
      COMMON /YMLV/ E,F(8),OLDF(8),X(10),XB(10)
      COMMON /STRESS/ S(10,8),ST(10)
C
C  GENERATE STRESS MATRIX.
C
      DO 10 I=1,10
      DO 10 J=1,N
   10 S(I,J) = 0.0
C
C  GENERATE NON-ZERO ELEMENTS.
C
      S(1,1) = E/360.0
      S(2,1) = -S(1,1)
      S(2,3) = -S(2,1)
      S(3,7) = S(1,1)
      S(4,5) = S(1,1)
      S(4,7) = -S(1,1)
      S(5,2) = -S(1,1)
      S(5,8) = S(1,1)
      S(6,4) = -S(1,1)
```

```
      S(6,6) = S(1,1)
      S(7,7) = 0.5*S(1,1)
      S(7,8) = S(7,7)
      S(8,1) = S(7,7)
      S(8,2) = -S(7,7)
      S(9,1) = -S(7,7)
      S(9,2) = -S(7,7)
      S(9,5) = S(7,7)
      S(9,6) = S(7,7)
      S(10,3) = S(7,7)
      S(10,4) = -S(7,7)
      S(10,7) = -S(7,7)
      S(10,8) = S(7,7)
C
      RETURN
      END
C ****************************************************************
      SUBROUTINE ESTRESS(IFLAG)
      COMMON /ORDER/ N,NOPTION
      COMMON /STIFFM/ SK(8,8),R(8)
      COMMON /STRESS/ S(10,8),ST(10)
      COMMON /YMLV/ E,F(8),OLDF(8),X(10),XB(10)
      DIMENSION XSIZE(10)
C
C COMPUTE ELEMENT STRESSES.
C
      DO 10 I=1,10
      ST(I) = 0.0
         DO 10 J=1,N
   10    ST(I) = ST(I) + S(I,J)*R(J)
C
      IF (IFLAG .EQ. 1) THEN
          DO 15 I=1,10
   15     XSIZE(I) = X(I)
      ELSE
          DO 18 I=1,10
   18     XSIZE(I) = XB(I)
      END IF
C
      WRITE(6,600)
  600 FORMAT(1H0,/,1H0,7X,'ELEMENT STRESSES AND SIZES',/,1H ,7X,26(1H-))
C
      DO 20 I=1,10
      WRITE(6,610) I,ST(I),I,XSIZE(I)
  610 FORMAT(1H ,'ST(',I2,' ) = ',F10.3,2X,'X(',I2,' ) = ',F10.3)
   20 CONTINUE
C
C COMPUTE WEIGHT.
C
      WEIGHT = 0.0
      WEIGHTD = 0.0
C
      DO 30 I=1,6
```

```
   30 WEIGHT = WEIGHT + XSIZE(I)
C
      DO 40 I=7,10
   40 WEIGHTD = WEIGHTD + XSIZE(I)
C
      WEIGHT = 0.1 * (360.0 * WEIGHT + 360.0 * SQRT(2.0) * WEIGHTD)
C
      WRITE(6,620) WEIGHT
  620 FORMAT(1H0,'WEIGHT = ',F10.3)
C
      RETURN
      END
C ***************************************************************
      SUBROUTINE LLT
      COMMON /ORDER/ N,NOPTION
      COMMON /YMLV/ E,F(8),OLDF(8),X(10),XB(10)
      COMMON /STIFFM/ SK(8,8),R(8)
C
C ROUTINE "LLT" PERFORMS THE CHOLESKI DECOMPOSITION OF
C THE COEFFICIENT MATRIX:
C
C                       SK = (L) * (LT)
C  WHERE,
C                        L = LOWER TRIANGULAR MATRIX
C                            (STORED IN LOWER TRIANGLE OF SK)
C
C                       LT = TRANSPOSE OF L
C
      DO 4 I=1,N
      DO 4 J=I,N
      S = SK(I,J)
      IF (I .EQ. 1) GOTO 2
      I1 = I-1
C
      DO 1 K=1,I1
    1 S = S - SK(I,K)*SK(J,K)
    2 IF (J .NE. I) GOTO 3
      IF (S .LE. 0.0) STOP 'NOT POSITIVE DEFINITE - STOP IN LLT'
      T = SQRT(ABS(S))
      SK(I,I) = T
      GOTO 4
    3 SK(J,I) = S/T
    4 CONTINUE
C
      RETURN
      END
C ***************************************************************
      SUBROUTINE FSUB
      COMMON /ORDER/ N,NOPTION
      COMMON /YMLV/ E,F(8),OLDF(8),X(10),XB(10)
      COMMON /STIFFM/ SK(8,8),R(8)
C
C ROUTINE "FSUB" PERFORMS THE FORWARD SUBSTITUTION TO SOLVE
```

```
C L*Y = B WHERE L IS STORED IN THE LOWER TRIANGLE OF A AND
C Y IS STORED IN THE X VECTOR.
C
      DO 10 I=1,N
      S=F(I)
      IM1 = I-1
      IF (I .EQ. 1) GOTO 10
C
        DO 20 J=1,IM1
   20   S = S - SK(I,J)*R(J)
C
   10 R(I) = S/SK(I,I)
C
C
      RETURN
      END
C *****************************************************************
      SUBROUTINE BSUB
      COMMON /ORDER/ N,NOPTION
      COMMON /YMLV/ E,F(8),OLDF(8),X(10),XB(10)
      COMMON /STIFFM/ SK(8,8),R(8)
C
C ROUTINE "BSUB" PERFORMS THE BACKWARD SUBSTITUTION TO SOLVE
C LT*R = Y WHERE L IS STORED IN THE LOWER TRIANGLE OF SK AND
C Y IS INITIALLY STORED IN X.
C
C ON RETURN R CONTAINS THE SOLUTION OF SK*R = F
C
C    WHERE,
C                 SK = THE ORIGINAL SYMMETRIC COEFFICIENT MATRIX.
C
      DO 3 I=1,N
      NI1 = N-I+1
      I1 = I-1
      S= R(NI1)
      IF (I .EQ. 1) GOTO 2
C
      DO 1 J=1,I1
    1 S = S - SK(NI1+J,NI1)*R(NI1+J)
    2 R(NI1)=S/SK(NI1,NI1)
    3 CONTINUE
      RETURN
      END
C *****************************************************************
      SUBROUTINE PRTANS
      COMMON /ORDER/ N,NOPTION
      COMMON /STIFFM/ SK(8,8),R(8)
C
C ROUTINE "PRTANS" PRINTS OUT THE SOLUTION VECTOR "X"
C
      DO 10 I=1,N
   10 WRITE(6,600) I,R(I)
  600 FORMAT(1H0,'R(',I2,' ) = ',F10.4)
```

```
C
      RETURN
      END
C ***************************************************************
      SUBROUTINE OPT3(NITER)
      COMMON /EPS/ TOL
      COMMON /YMLV/ E,F(8),OLDF(8),X(10),XB(10)
C
      WRITE(6,600)
  600 FORMAT(1H0,'WHAT NUMBER OF ITERATIONS WOULD YOU LIKE......:')
      READ(5,*) NITER
C
      WRITE(6,610)
  610 FORMAT(1H0,'WHAT TOLERANCE DO YOU REQUIRE OF THE',
     +           'SOLUTION VECTOR................................:')
      READ(5,*) TOL
C
      RETURN
      END
C ***************************************************************
      SUBROUTINE SETUP
      COMMON /ELEMAT/ EK(8,8,10)
      COMMON /ORDER/ N,NOPTION
      COMMON /STIFFM/ SK(8,8),R(8)
      COMMON /YMLV/ E,F(8),OLDF(8),X(10),XB(10)
      DIMENSION DELTAK(8,8),TEMP(8)
C
C COMPUTE "DELTA K" MATRIX.
C
      CALL GENDK(DELTAK)
C
C COMPUTE THE RIGHT HAND SIDE OF EQ.(17)
C
C FIRST MULT. [DELTA K] * (R)
C
      DO 30 I=1,N
      TEMP(I) = 0.0
           DO 30 J=1,N
   30      TEMP(I) = TEMP(I) + DELTAK(I,J) * R(J)
C
      DO 40 I=1,N
   40 F(I) = OLDF(I) - TEMP(I)
C
      RETURN
      END
C ***************************************************************
      SUBROUTINE ITER(NITER)
      COMMON /EPS/ TOL
      COMMON /STIFFM/ SK(8,8),R(8)
      DIMENSION OLDR(8)
C
C THIS ROUTINE PERFORMS THE DEFLECTION & STRESS ANALYSIS VIA
C SIMPLE ITERATION.
```

```
C
      DO 10 I=1,NITER
C
C SET UP THE RHS OF EQ.(17)
C
      CALL SETUP
C
C  SAVE THE OLD SOLUTION VECTOR 'R' IN 'OLDR'.
C
      DO 20 J=1,8
   20 OLDR(J) = R(J)
C
C SOLVE THE SYSTEM FOR A NEW "RHS".
C
C FIRST CALL THE FORWARD SUBSTITUTION PART....
C
      CALL FSUB
C
C ...NOW USE THE BACKWARD SUBSTITUTION.
C
      CALL BSUB
C
C TEST TO DETERMINE IF MAGNITUDE OF "OLD" SOLUTION VECTOR
C IS WITHIN A GIVEN TOLERANCE OF THE "NEW" SOLUTION VECTOR.
C
      DO 30 J=1,8
      DIFF = ABS(OLDR(J) - R(J))
      IF (DIFF .LT. TOL) GOTO 30
      GOTO 10
   30 CONTINUE
      WRITE(6,620) I
  620 FORMAT(1H0,'THE ANALYSIS REQUIRED ',I2,' ITERATIONS.')
      GOTO 40
C
   10 CONTINUE
      WRITE(6,620) NITER
C
C PRINT OUT THE ANSWERS...
C
   40 CALL PRTANS
C
C PRINT OUT THE ELEMENT STRESSES AND SIZES.
C
      CALL ESTRESS(2)
C
      RETURN
      END
C ****************************************************************
      SUBROUTINE CHGSIZE(IFLAG)
      COMMON /YMLV/ E,F(8),OLDF(8),X(10),XB(10)
C
C THIS ROUTINE ALLOWS THE USER TO MODIFY THE STRUCTURE.
C
```

```
    5 WRITE(6,600)
  600 FORMAT(1HO,'HOW MANY MEMBER SIZES WOULD YOU LIKE TO CHANGE....:')
      READ(5,*) NCHGES
      IF (NCHGES .GT. 10) GOTO 5
      IF (NCHGES .LE. 0) RETURN
C
      IF(IFLAG.EQ.1) THEN
C
C COPY EXISTING MEMBER SIZES TO MODIFIED MEMBER.
C
      DO 8 I=1,10
    8 XB(I) = X(I)
      END IF
C
      DO 10 I=1,NCHGES
   15 WRITE(6,610)
  610 FORMAT(1HO,'ELEMENT NUMBER........:')
      READ(5,*) NUMELEM
C
C CHECK TO DETERMINE IF MEMBER SIZE VALID.
C
      IF (NUMELEM .LT. 0 .OR. NUMELEM .GT. 10) THEN
           WRITE(6,900) NUMELEM
  900      FORMAT(1HO,' *** NO SUCH ELEMENT NUMBER --> ',I3,' ***')
           GOTO 15
      END IF
      WRITE(6,620)
  620 FORMAT(1H ,'MEMBER SIZE..........:')
      READ(5,*) SIZEMEM
   10 XB(NUMELEM) = SIZEMEM
C
      RETURN
      END
C *****************************************************************
      SUBROUTINE GENDK(DELTAK)
      COMMON /ELEMAT/ EK(8,8,10)
      COMMON /ORDER/ N,NOPTION
      COMMON /YMLV/ E,F(8),OLDF(8),X(10),XB(10)
      DIMENSION DELTAK(8,8)
C
C INITIALIZE THE MATRIX "DELTAK" TO ZERO.
C
      DO 5 I=1,N
           DO 5 J=1,N
    5      DELTAK(I,J) = 0.0
C
C COMPUTE DELTA K (EQ. (18))
C
      DO 10 I=1,10
      DIFFX = XB(I) - X(I)
C
          DO 20 J=1,N
               DO 20 K=1,N
```

```
   20          DELTAK(J,K) = DIFFX * EK(J,K,I) + DELTAK(J,K)
   10      CONTINUE
C
        DO 23 I=1,N
            DO 23 J=1,N
   23       DELTAK(I,J) = (E/360.) * DELTAK(I,J)
C
        RETURN
        END
C ***************************************************************
        SUBROUTINE OPT2
        COMMON/ELEMAT/ EK(8,8,10)
        COMMON/YMLV/ E,F(8),OLDF(8),X(10),XB(10)
        COMMON/STIFFM/ SK(8,8),R(8)
        DIMENSION ROLD(8),DELR(8)
C
C  STORE OLD R VECTOR
C
        DO 10 I=1,8
   10 ROLD(I)=R(I)
C
C INITIALIZE DELR=0.
C
        DO 20 I=1,8
   20 DELR(I)=0.0
C
C  LOOP FOR SUMMATION, EQ. (15)
C
        DO 200 J=1,10
C
C  FORM RIGHTHAND SIDE OF EQ. (16)
C
        DO 120 I=1,8
          F(I)=0.0
          DO 110 L=1,8
  110     F(I)=F(I)+EK(I,L,J)*ROLD(L)
  120 F(I)=-F(I)*E/360.
C
C  SOLVE EQ. (16) FOR R
C
        CALL FSUB
        CALL BSUB
C
C  SUM DELR
C
        DO 130 I=1,8
  130 DELR(I)=DELR(I)+(XB(J)-X(J))*R(I)
C
  200 CONTINUE
C
C  SOLVE EQ. (15)
C
        DO 210 I=1,8
```

```
  210 R(I)=ROLD(I)+DELR(I)
C
C  PRINT OUT SOLUTION VECTOR
C
      CALL PRTANS
C
C  COMPUTE ELEMENT STRESSES
C
      CALL ESTRESS(2)
C
C  REPLACE NEW R VECTOR WITH ORIGINAL
C  (RETURN TO ORIGINAL STRUCTURE)
C
      DO 220 I=1,8
  220 R(I)=ROLD(I)
      RETURN
      END
```

REFERENCES

1. Taylor, R. F., "Automated Design of Damage Resistant Structures, Vol. I - Theory and Applications," AFWAL-TR-81-xxxx (in progress), July 1982.
2. Kloos, G. R., "Cross Reference Program (XREF)," University of Dayton Research Institute Report (support documentation for UDR-TR-81-44), July 1981.
3. Venkayya, V. B. and Tischler, V. A., "OPTSTAT - A Computer Program for the Optimal Design of Structures Subjected to Static Loads," AFFDL-TM-FBR-79-67, June 1979.

* U.S. GOVERNMENT PRINTING OFFICE: 1983 - 659-062/766